МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет кораблебудування
імені адмірала Макарова

В.А. ТУДОРАН

ЛЮДИНО-МАШИННА ВЗАЄМОДІЯ

Навчальний посібник

Рекомендовано Методичною радою НУК

Миколаїв ◊ НУК ◊ 2013

УДК 004.5(075.8)
ББК 73я73
Т 81

Автор:
В. А. Тудоран, старший викладач

Рецензенты:
П. І. Жежнич, доктор технічних наук, доцент
К.В. Кошкін, доктор технічних наук, професор
В. І. Ситніков, доктор технічних наук, професор

Рекомендовано Методичною радою НУК
як навчальний посібник

Тудоран В. А.

Т 81 Людино-машина взаємодія / В. А. Тудоран. – Миколаїв : НУК, 2013. – 180 с.

Навчальний посібник містить теоретичний матеріал, питання для самоконтролю, список літератури, яка необхідна при вивченні курсу, а також основні визначення, властивості людино-машинної взаємодії, психологічні принципи взаємодії людини й програмного забезпечення, основні функціональні елементи інтерфейсу, рекомендації з проектування й прототипування інтерфейсу, огляд засобів розробки людино-машинного інтерфейсу.

З метою кращого засвоєння теоретичного матеріалу запропоновані практичні завдання й завдання з лабораторних робіт, доповненні рекомендаціями й прикладами їх розв'язку.

Призначено для студентів напрямків «Програмна інженерія» й «Комп'ютерні науки», а також має бути корисним для студентів, аспірантів, професійна діяльність яких пов'язана з розробкою програмного забезпечення.

УДК 004.5(075.8)
ББК 73я73

імені адмірала Макарова, 2013

ЗМІСТ

5.9. Ментальні моделі.

5.10 Засоби підтримки користувача

5.11 Що мотивує людину

Тема 6. Функціональні компоненти

6.1. Популярні стилі користувальницького інтерфейсу

6.2. Первинні й вторинні вікна

6.3. Піктограми

6.4. Меню

6.5. Кнопки

6.6. Функціональність клавіатури

6.7. Правила взаємодії з об'єктом

6.8. Організація пошуку

6.9. Запрошення

6.10. Основні функціональні елементи веб-інтерфейсів

Тема 7. Візуальна організація. Використання звуків і анімації.

7.1. Основи розмітки сторінки

7.2. Візуальний плин: Що мені варто подивитися далі?

7.3. Підбор екранних шрифтів

7.4. Колірне коло: застосування. Колірні гармонії

7.5. Зниження шуму на сторінці

7.6. Звуки й інтерфейс.

7.7. Анімація в інтерфейсі

7.8 Особливості проектування для мобільних пристроїв.

Тема 8. Оцінка якості людино-машинного інтерфейсу. Юзабіліті-тестування.

8.1. Кількісна оцінка інтерфейсу.

8.2. Якісна оцінка інтерфейсу

Тема 9. Засоби розробки людино-машинного інтерфейсу

Частина 2. ПРАКТИКУМ ЗА КУРСОМ "ЛЮДИНО-МАШИННА ВЗАЄМОДІЯ"

Практичні завдання для закріплення теоретичного матеріалу з прикладами виконання

Лабораторні роботи з рекомендаціями

Лабораторна робота № 1

Лабораторна робота № 2

Лабораторна робота № 3

Лабораторна робота № 4

Лабораторна робота № 5

КОНТРОЛЬНІ ПИТАННЯ

СПИСОК ЛІТЕРАТУРИ

Вступ

Якщо водій, сідаючи за кермо незнайомого автомобіля, знає, куди вставити ключ запалювання, і де перебувають педалі газу й гальма, то користувачі нових програмних продуктів до останнього часу були позбавлені подібної переваги.

Автомобілебудівники спираються на багаторічні традиції, головний зміст яких - «усе на благо водія». Отут і ремені безпеки, і дзеркала заднього виду, і, нарешті, кондиціонер у салоні.

Розробники програм усе ще орієнтуються в основному на особистий досвід і, здебільшого, ставляться до користувачів зневажливо.

І лише в останні роки, коли число власників персональних комп'ютерів стало наближатися до числа автомобілістів, ситуація трохи змінилася.

Зміни полягають у тім, що творці програмних продуктів стали намагатися, по-перше, поставити себе на місце потенційного користувача, і, по-друге, уніфікувати «педалі».

Першим хто задумався про взаємодію людини й комп'ютера, засновану на особливостях людського мислення й поводження був Ванневар Буш, американський інженер і розробник аналогових комп'ютерів (1890 - 1974). В 1945 році він написав книгу «Як ми можемо думати» («As We May Think»), де представив систему Memex (від англ. memory - пам'ять, index - індекс). Ідеї Буша, застосовані у цій машині, стали основою для багатьох технологій в еру цифрових комп'ютерів - гіпертекст, інтернет, розпізнавання мови, онлайн енциклопедії.

Суттєвим кроком у напрямі зростання зручності взаємодії людини й комп'ютера був перехід від систем, що займали цілі приміщення, з безліччю тумблерів, перемикачів, й інших елементів керування, які повинні були обслуговуватися декількома фахівцями, до настільного комп'ютера.

Наступним етапом розвитку стала поява графічного інтерфейсу, коли можна безпосередньо управляти елементами інтерфейсу за допомогою

пристроїв уведення/виводу, наприклад світлового пера, що робило інтерфейс більше зрозумілим.

9 грудня 1968 року комп'ютерна миша була представлена на демонстрації інтерактивних пристроїв у Каліфорнії Дугласом Карлом Енгельбартом (1903 - 1993). Енгельбарт використав мишу для керування вказівником на екрані. У той час про комп'ютери думали як про безособові коробки, які читають перфокарти, шумлять якийсь час і видають стопки паперу. Графічне керування в режимі реального часу було просто неймовірним. Першим комп'ютером, у комплект якого включалася миша, був мінікомп'ютер Xerox 8010 Star Information System (*англ.*), представлений в 1981 році. Миша фірми «Xerox» мала три кнопки й коштувала 400 доларів США. В 1983 році фірма «Apple» випустила свою власну модель однокнопкової миші для комп'ютера Lisa, вартість якої вдалося зменшити до 25 доларів. Широку популярність миша придбала завдяки використанню в комп'ютерах Apple Macintosh і пізніше в операційній системі (ОС) Windows для IBM PC сумісних комп'ютерів.

Неоціненний внесок у розвиток графічного інтерфейсу вніс Алан Кей, який сьогодні є президентом дослідницького інституту В'юпоінта. Він запропонував концепцію Dynabook, що визначила концептуальну базу для ноутбуку, планшетного комп'ютеру та електронної книги, й є архітектурою сучасного віконного графічного інтерфейсу. На початку 1972 року в ній втілилися всі елементи графічного інтерфейсу користувача, які ми маємо сьогодні: робочий стіл, папки, вікна.

Популярність ідея віконного інтерфейсу одержала більше ніж через двадцять років у Windows'95. Такий самий період потрібен був для популяризації миші. У своїй теорії довгого носа, Білл Бакстон говорить, що те, що буде винайдено через десять років, уже сьогодні є ідеєю 10-річної давнини. Тому перш ніж винаходити нове подивиться навколо й використайте те, що є.

Інтерфейс користувача, взаємодія між користувачем і комп'ютером вимагає ретельної розробки. Інтерфейс це те, що продається. Нікого не цікавить, як улаштований автомобіль поки він працює. Користувач не бачить

геніального коду або новітніх технологій, а тільки інтерфейс, саме за нього він платить гроші. Комп'ютер повинен полегшити життя людині, а не ускладнити його, як це відбувається з банкоматами по усьому світі. Про те, що ви ввели невірний пін-код або в банкоматі недостатньо грошей, ви довідаєтеся тільки після того, як відповісте в яких купюрах ви хочете одержати гроші, чи друкувати чек, чи точно ви хочете друкувати чек або бажаєте берегти ліс, чи не хочете ви пожертвувати на благодійність, і т.д. А черга позаду вас у цей час зітхає й нервує. Зручний інтерфейс – це постійні клієнти й їхні рекомендації. Незадоволений користувач покине вас, як тільки з'явиться альтернатива. Задоволений простить навіть невдалі продукти, як неодноразово показує приклад компанії «Apple».

Для проектування зручного інтерфейсу важливо розібратися, яки властивості йому притаманні, а також існуючи методи, засоби та рекомендації щодо його проектування й розробки

Частина 1. ПОНЯТТЯ, ПРИНЦИПИ, ВЛАСТИВОСТІ Й ПРОЕКТУВАННЯ ЛЮДИНО-МАШИННОЇ ВЗАЄМОДІЇ

Тема 1. Поняття інформаційної взаємодії

1.1. Важливість проектування інтерфейсу

Щоб стати хорошим програмістом, необхідно співчутливо ставитися до природи й потреб комп'ютера. Однак природа й потреби комп'ютера зовсім далекі природі й потребам людської істоти, якій доводиться в остаточному підсумку цей комп'ютер використовувати. Створення програмного забезпечення вимагає таких інтелектуальних зусиль, так поглинає програмістів, що вони мають повністю поринати в далекий людині розумовий процес. Для програміста потреби процесу програмування одержують пріоритет перед будь-якими потребами користувачів із зовнішнього світу, і більше того – навіть мови цих двох світів конфліктують.

Ключ до рішення проблеми створення зручних інтерфейсів - проектування взаємодії. І займатися цим повинні люди, які в розробці власне програмного забезпечення участі не приймають. Їхнє завдання зрозуміти як людина думає в рамках заданої предметної області, при взаємодії з конкретним видом устаткування, урахувати попередній досвід потенційних користувачів і спроектувати поводження комп'ютера. Сьогодні програмісти свідомо проектують «код» програм, але лише ненавмисно проектують взаємодію з людьми. Вони проектують можливості, але не те, як цими можливостям користуватися – як програма поводиться, спілкується, або повідомляє. Тому проектувальники взаємодії повинні зосередитися безпосередньо на тім, як користувачі сприймають продукти, засновані на програмному забезпеченні.

На жаль, в області проектування взаємодії було загублено багато часу. Технічне й програмне забезпечення за своїми можливостями пішло далеко вперед і давно дозволяє забезпечити максимум підтримки користувачеві. Проте, користувач дотепер змушений підладжуватися під програми.

Потрапивши на незаселений острів, людина не стане заперечувати, якщо корабель, що прийшов рятувати, виявиться іржавим кістяком, поїденим течами й повним пацюків. Різниця між наявністю програмного рішення проблеми й відсутністю рішення взагалі настільки велике, що ми приймаємо будь-які випробування й труднощі, що супроводжують це рішення.

Непіддатливість проблеми відбувається не через складність створення більш досконалих взаємодій. Вона походить із нашої загальної готовності приймати неякісні взаємодії як неминуче. Побачивши проіржавілий корабель, ми не цікавимося, які на ньому зручності, а просто стрибаємо на борт і щасливі тим, що одержали.

Та ситуація, у якій сьогодні перебуває розробка користувацьких інтерфейсів має наступні причини:

1. З комп'ютерами працювало вузьке коло людей, всі незручності – це їхня професія;
2. Первісно комп'ютери були малопотужні, що змушувало програмістів працювати під девізом «Будь добрий до мікросхем і безжалісний до користувача», автор якого Білл Могридж (Bill Moggridge), промисловий дизайнер, творець першого ноутбука;
3. Програмісти пишуть орієнтуючись на себе, а не на звичайних людей. При розробці програми людина думає мовою комп'ютера, на такій же мові розробляється інтерфейс. Після того як розробник провозився два тижні зі своєю програмою – йому в ній все повністю звично й зрозуміло, і важко подивитися з іншого боку;
4. Через відсутність кращого користувачі звикли миритися з тим, що є.

Сьогодні багато чого змінилося. Комп'ютери стали для кожного. Потужності досить для підтримки будь-якого інтерфейсу, тому кредо програмістів «Будь добрий до мікросхем і безжалісний до користувачів» уже не актуально. Навіть з'явилися фахівці із проектування взаємодії. Однак ці нові віяння торкнулися, насамперед, ведучих фірм-розробників програмного забезпечення (ПО), таких як Apple, Microsoft, і практично не вплинули на стиль

роботи програмістів-одинаків або невеликих колективів. Хоча зазначені категорії розроблювачів ПЗ користуються досить обмеженим набором інструментальних засобів, створювані ними програмні продукти досить помітно розрізняються за організацією взаємодії з користувачем. При цьому розходження проявляються як на рівні зовнішнього оформлення інтерактивних компонентів додатка, так і на рівні принципів (поглядів розроблювача), покладених в основу реалізації цих компонентів.

У більшості фірм, що займаються розробкою програмного забезпечення, просто немає людей, що мають уявлення про проектування для кінцевого користувача. Але є люди, які мають більш ніж серйозне уявлення про проектування програм, свою власну думку й особисті переваги.

1.2. Інтерфейс користувача - міст між людиною й комп'ютером

Користувацький інтерфейс — це сукупність інформаційної моделі проблемної області, засобів і способів взаємодії користувача з інформаційною моделлю, а також компонентів, що забезпечують формування інформаційної моделі в процесі роботи програмної системи.

Під *інформаційною моделлю* розуміється умовне подання проблемної області, що формується за допомогою комп'ютерних (візуальних і звукових) об'єктів, які відображують склад і взаємодію реальних компонентів проблемної області.

Термін «проектування взаємодії» переважний терміну «проектування інтерфейсу», оскільки слово «інтерфейс» припускає, що код перебуває в одному місці, люди в іншому, а інтерфейс між ними дозволяє обмінюватися повідомленнями. Основна перевага хорошого інтерфейсу користувача полягає в тім, що користувач завжди почуває, що він управляє програмним забезпеченням, а не програмне забезпечення управляє ним.

Мається на увазі, що саме інтерфейс відповідає за потреби користувачів. Щоб забезпечити користувачів відчуттям могутності й задоволення, необхідно

спочатку думати концептуально, потім у термінах поводження й лише в останню чергу – у термінах інтерфейсу.

1.3. Рівні складності й орієнтація на користувача

Користувачів турбує набагато менша кількість речей, ніж думають розробники. Користувачі застосовують програми для рішення своїх завдань. І все, що їх хвилює - це розв'язання цих завдань. Якщо мова йде про графічну програму, вони захочуть контролювати кожний окремий піксель, щоб досягти найвищої чистоти деталей. Якщо в їхніх руках програма для виготовлення веб-сторінок, вони ляжуть кістьми для того, щоб їхня сторінка виглядала точно так, як вони собі уявляють.

Але користувачі не задумаються про те, чи перебуває панель інструментів угорі або внизу екрану, чи індексовані розділи допомоги чи ні. Вони байдужні до багатьох речей. І відповідальність дизайнерів саме в тім і полягає, щоб звільнити користувачів від необхідності приймати подібні рішення. Перенесення відповідальності за прийняття подібних рішень на користувача – недоробка програміста, який не потурбувався додумати до кінця, яке ж рішення в цьому випадку буде оптимальним.

Для користувача не має значення, який саме процесор використано й чи є мова програмування об'єктно-орієнтованою, додаток багатопоточним, або називається якимись іншими розумними словами. Для користувачів важливіше за все зручність і результати. Проте все, що вони бачать – це інтерфейс. Інакше кажучи, з погляду споживача саме інтерфейс є кінцевим продуктом.

Розробникам подобається ділити користувачів за групами: непідготовлений користувач, комп'ютерно-освічений і просунутий.

«Непідготовлений користувач» – позначення користувача дурного й некомпетентного. Причина того, що користувачі почувають себе дурними й некомпетентними, не їхні особисті якості, а неякісне проектування взаємодії. «Комп'ютерно-освічений користувач» це така ж сама людина, якій так багато разів робили боляче, що товщина рубцевої тканини вже просто не дозволяє їй

відчувати біль. Освічений користувач у разі втрати документу програмою не панікує, а починаєте повільний, самостійний, зовсім непотрібний пошук файлу в ієрархічній файловій системі, і при цьому не скаржитеся. Немає особливого змісту постійно нити й скаржитися на програму, що становить обов'язкову й постійну частину вашої роботи. Більшість людей навіть не усвідомлює своїх титанічних зусиль, спрямованих на компенсацію недоліків інструментів, заснованих на програмному забезпеченні.

Просунуті користувачі – це техноентузіасти, які змогли придушити свої інстинкти, щоб стати корисними споживачами продукції з високим опором природній людській логіці. Вони пишаються випробуваннями, як альпіністи на Евересті.

Такі люди стверджують, що підготовка є необхідною для користування комп'ютером, тому що підготовка й складання іспиту на знання правил дорожнього руху потрібно для водіння автомобілю. Однак вони випускають із уваги, що помилка при водінні автомобілю часто приводить до загибелі людей, тоді як помилка при роботі із програмою має звичайно менш суворі наслідки. Люди вчилися б водити так само, як освоюють Excel, якщо б машини не були настільки смертоносними.

Інтерфейс – це дещо більше, ніж вікна, піктограми, різні види меню й миша. Незалежно від того, наскільки складним є задача, що розв’язується тим або іншим продуктом, складові частини цієї задачі однаково повинні залишатися простими.

Спосіб, яким людина виконує будь-яке завдання за допомогою будь-якого продукту, а саме чинені нею дії й те, що людина одержує у відповідь, і є інтерфейсом.

Тема 2. **Аналіз, проектування й прототипування людино-машинного інтерфейсу**

2.1 Життєвий цикл програмного забезпечення

Після закінчення майже двох десятків років активного використання комп'ютерних технологій Міністерство оборони США дійшло висновку, що основною причиною невдач великих інформаційних проектів є методологія їхньої реалізації (точніше, її відсутність). Від неприємних сюрпризів не могли врятувати ні досвідчені менеджери, ні вичерпне тестування, ні застосування CASE-засобів. Кожна нова виявлена помилка змушувала майже повністю переглядати проект. І навіть, якщо зрештою вдавалося одержати припустимий результат, при надходженні наступного замовлення картина повторювалася. Усвідомлення зазначених обставин змусило Міністерство оборони США зробити крок у напрямку стандартизації процесів розробки програмних систем. Перший стандарт був затверджений в 1985 році, а в 1994-1996 роках був розроблений і прийнятий новий, значно більше детальний і глибокий стандарт - MIL-STD-498. Він являє собою комплект із трьох документів загальним обсягом близько 600 сторінок і встановлює термінологію, процеси, завдання й об'єкти, використовувані при розробці й супроводі проектів програмних систем. Основні положення цього військового стандарту погоджені з міжнародним стандартом ISO/IEC 12207:1995 («Процеси життєвого циклу програмних засобів»), але разом з тим є більше конкретними й наближеними до практики.

У нашій країні спроба стандартизації процесів створення програмних систем вилилася у створення комплекту документів під загальною назвою «Єдина система програмної документації», який побачив світ ще в 1977 році й періодично коректувався аж до останнього часу.

При досить помітному розходженні в деталях всі відомі стандарти використовують у якості вихідного положення поняття життєвого циклу програмного продукту (або системи).

Під *життєвим циклом* розуміють послідовність процесів, дій і завдань, які здійснюються в ході розробки, експлуатації (використання) і супроводу програмного продукту протягом всього його життя, від визначення вимог до завершення використання.

Практично всі стандарти (навіть військові) передбачають можливість їхньої адаптації до особливостей конкретного проекту за умови дотримання основних вимог до технології й показників якості продукту. Наприклад, на етапі формування вимог до системи повинні враховуватися: область застосування системи, вимоги користувача (замовника) до функціональних можливостей системи, до рівня її безпеки й захищеності; ступінь пристосованості до фізичних і психологічних особливостей користувача; вимоги до рівня кваліфікації користувачів; ступінь документованості системи; організація супроводу й т. п.

Необхідно підкреслити, що положення стандартів залишаються справедливими незалежно від призначення, рівня складності й складу колективу розробників програмного продукту, тобто навіть у тих випадках, коли мова йде про невелику програмну утиліту, а колектив розробників складається з однієї людини.

Основними етапами розробки інтерфейсу є аналіз, прототипування й проектування.

2.2 Аналіз

Керуючими аспектами при проектуванні інтерфейсу повинні бути завдання, цілі й цінності потенційних користувачів. Це можливо тільки при тісній роботі з користувачами протягом усього процесу розробки. Прийняття остаточних рішень повинне ґрунтуватися на точці зору користувачів, їхній роботі й оточенні. При проектуванні потрібно прагнути спиратися на думку реальних користувачів, а не їхніх начальників, які мають приблизне уявлення про те, як у дійсності відбувається робота. Для збору даних для аналізу існує декілька методів.

Одним з основних, що дозволяють зібрати максимум інформації є *спостереження*. Спостереження застосовується там, де втручання експериментатора порушить процес взаємодії людини із середовищем, адже, як відомо, експеримент проводиться в штучно створених умовах. Цей метод незамінний у випадку, коли необхідно одержати цілісну картину того, що відбувається. Наприклад, як відбувається робота користувача до впровадження розроблювального програмного забезпечення, які дії він звичайно робить, як часто вони повторюються, з якими проблемами користувач зіштовхується, яке програмне забезпечення застосовує на даний момент, які завдання воно вирішує, до яких технічних засобів користувач має доступ.

Спостереження пов'язане з деякими труднощами:

1. Об'єктивними (не залежать від спостерігача):
   - обмежений і приватний характер кожної спостережуваної ситуації;
   - не всі ситуації піддаються спостереженню;
   - поводження користувача може відрізнятися від природного;
   - складно забезпечити репрезентативність;
   - висока трудомісткість (обробці піддається великий обсяг інформації).
2. Необ'єктивні (пов'язані з особистісними й професійними особливостями):
   - автоморфізм (схильність витлумачувати дії інших через власне «я»);
   - на якість інформації може впливати відмінність соціального стану, досвіду спостерігачів і тих, хто спостерігає;
   - зміна поводження користувача під впливом очікувань спостерігача;
   - труднощі інтерпретації (не завжди форми поводження й мотиви зв'язані однозначно).

Попередньо необхідно підготувати програму спостереження. Для цього визначити:

- об'єкт дослідження (індивід, мала група, соціальна спільність, захід, подія);
- предмет спостереження (сукупність аспектів поводження користувача і його взаємодій з використовуваними ним пристроями, які цікавлять спостерігача);
- спостережувану ситуацію;
- категорії спостереження (показники або ознаки, які відповідають певним вимогам);
- умови спостереження;
- одиниці спостереження.

Ще один метод збору даних – *інтерв'ю*. Інтерв'ю – вид бесіди між двома або більшою кількості людей, при якій інтерв'юер задає питання своїм співбесідникам й отримує на них відповіді.

Перший крок – підбір учасників. Це може бути репрезентативна або цільова група користувачів, розмова з ними є найбільш важливою. Якщо створюється автоматизована система навчання для лекційних курсів, виникає питання, яку ж групу вибрати цільовою: викладачів, студентів, їхніх батьків, співробітників міністерства? До того ж студенти, які будуть користуватися цією системою можуть навчатися як на денному так і на заочному відділенні. Якщо взяти по одній людині з кожної такої групи, то буде досягнута повна репрезентативність.

У випадку, коли розробник прагне створити щось краще за існуючі аналоги, то треба поговорити з користувачами подібних програмних продуктів, щоб з'ясувати з якими проблемами вони зіштовхуються. Якщо у нього немає можливості поговорити ні з тими ні з іншими, можна навіть провести інтерв'ю з людьми, які не мають відношення до продукту, але можуть собі уявити про що йдеться мова. Головне, щоб вони не брали участь у його розробці. Наприклад, це можуть бути друзі й родичі розробника. Звичайно, в останньому випадку треба бути більше обережним з виводами, але навіть приблизна репрезентативна група краще, ніж нічого.

Другий крок – це правильна постановка питань. Наприклад: «чи є щоденне оновлення важливим для вас?». Таке питання є навідним. Воно спонукує людину відповісти: «Так, звичайно, чому ні?». Ефективніше було б запитати: «Я помітив, що ви не користуєтеся/користуєтеся оновленнями. Чому?». Таке питання є відкритим і змушує людину задуматися про те, наскільки оновлення дійсно важливі для нього.

Варто уникати наступних типів питань:

- що б вам хотілося (сподобалося) в гіпотетичному сценарії (звичайно люди або бояться скривдити й їм усе подобається, або люблять критикувати й їм усе не подобається. Особливо варто уникати подібних питань щодо елементів інтерфейсу, тому що самі по собі вони можуть бути прекрасним, але в конкретному додатку зовсім марними);
- як часто ви робите... (наприклад, гімнастику. Звичайно, більшість людей відповість, що часто. У цьому випадку краще конкретизувати питання – скільки разів ви робили гімнастику минулого тижня. А скільки цього?);
- на скільки балів ви оцінюєте щось за абсолютною шкалою;
- питань, що вимагають бінарних відповідей (так/ні). Однаково потім буде потрібно задавати уточнюючі питання.

Після того як питання задано, треба дати людині можливість відповісти. Тобто помовчати якийсь час поки вона не почне говорити.

Кращі з кращих дизайнерів користувацьких інтерфейсів одностайні в одному: перед тим як розробляти дизайн інтерфейсу, потрібно продумати й описати його передбачуваних користувачів.

Це потрібно для того, щоб не зосереджуватися на проблемах конкретного користувача, а виявити загальні завдання. На основі проведених інтерв'ю й спостережень виводиться середньостатистичний образ представника тієї або іншої цільової групи. Такий образ називають *профілем користувача* (або *персонажем*). Краще, якщо таких профілів буде не більше трьох, тому що задовольнити тисячу користувачів однаково не можливо. Тому краще задовольнити одного на 100%, чим 100 користувачів на 1%. У першому

випадку задоволений користувач усім розповість про прекрасну програму, у другому всі просто обміняються негативними відгуками й звернуться до ваших конкурентів. Кожний профіль бажано олюднити – дати йому ім'я, наділити особистими особливостями й рисами характеру, й за можливістю знайти відповідне фото. Подивившись на нього при розробці інтерфейсу, відразу стає зрозуміло, буде він користуватися запропонованими варіантами інтерфейсу чи ні.

2.3 Проектування

Проектування **–** процес створення праобразу майбутнього інтерфейсу, та способів його виготовлення. Задача проектування – встановити структуру інтерфейсу (кількість й призначення екранних форм), встановити типи зв'язків (умови переходів між екранними формами), визначити атрибути (функціональні вимоги до кожної форми). Проект інтерфейсу виходить з виводів, зроблених на основі аналізу. Опис концепцій впливає на їхню реалізацію в системі, особливо в ситуаціях, коли якість навчання роботі з системою є найважливішої складовою загальної якості. Це спостереження цілком підтверджується досвідом. Наприклад, Джеф Раскін, батько Macintosh, спочатку був начальником відділу документації. Після того, як він виявив, що розроблену систему неможливо зрозуміло описати, він створив нову, яка описується добре. Побічною властивістю нової системи, комп'ютера Макінтош, було те, що його інтерфейс був зрозумілий і зручний у роботі. Документація є частиною інтерфейсу, при чому в складних системах – більша його частина.

Етапи створення програмного забезпечення можна порівняти з етапами створення кінофільму (Таб. 1). Якщо режисер вирішить замість поїзда підірвати літак на етапі підготовки, то просто перепишуть сценарій. Якщо пропустити цей етап, то час на роздум збіжиться із часом виробництва й змінити рішення в той час, коли поїзд уже стоїть й чекає на вибух, обійдеться значно дорожче. Те ж саме стосується й створення програм.

Таблиця 1 Порівняння етапів створення кіно й програм

Підготовка	Виробництво	Доведення
Кінофільми		
Написання сценарію, розкадрування, створення декорацій, підбір акторів, пошук фінансування.	Включаються камери, режисер кричить «Зняте», горять софіти, актори грають, гримери рум'янять	Монтаж, озвучення, просування на ринку
Програми		
Проектування взаємодії, розробка сценаріїв, прототипування інтерфейсу, наймання програмістів, пошуки грошей.	Програмісти пишуть код, менеджери бігають за піцою, проектувальники вирішують дрібні проблеми взаємодії	Налагодження, документування, просування на ринку.

Розхожа істина про розробку програмного забезпечення говорить, що досягти якісної взаємодії можна тільки поетапним поліпшенням. Прихильність юзабіліті-тестуванню в багатьох великих компаніях, зокрема в Microsoft, привела до поширення цієї ідеї. Звичайно, ітерації - важливий елемент якісного проектування: продовжуємо працювати, поки не одержимо правильне рішення. Однак багато розробників зрозуміли ідею інакше: плюємо на проектування й просто перераховуємо в темряві всі купини, які є на дорозі.

В 1986 році компанія «Microsoft» поквапилася з виходом на ринок першої версії Windows, яка була настільки сміховинна, що заслужено стала предметом жартів. Шість місяців потому «Microsoft» випустила версію 1.03 і виправила деякі дефекти. Роком пізніше Microsoft випустила версію 1.1, а потім версію 2.01. На кожному етапі розвитку продукту розробники намагалися розв'язати проблеми, створені в попередній версії. Нарешті, чотири роки потому після випуску першої версії, «Microsoft» представила Windows 3.0, і всі перестали сміятися.

Мало компаній в цій індустрії мають таку завзятість і фінансові можливості, що дозволяють витримати чотири роки публічного приниження й досягти, нарешті, приємного результату. При цьому всі спостерігають, як лідер

дефакто сліпо спотикається практично до переможного кінця, після чого роблять очевидний вивід про те, що саме так і потрібно робити.

Стратегія змору ефективна, тільки якщо застосовується компаніями, які володіють залізобетонним ім'ям, купою часу, витримкою гравця в покер і невичерпними фінансами. Дотепер жоден учасник комп'ютерної індустрії не виявив всі ці якості на рівні, який відповідає рівню «Microsoft».

Друга основна помилка, чинена при проектуванні – це сліпо додержуватися думки клієнта. У вісімдесяті роки «IBM» дуже пишалася статусом компанії, веденої клієнтами. Тоді «IBM» володіла практично всім комп'ютерним бізнесом, у масштабах більше серйозних, чим сьогодні «Microsoft», однак зараз, залишаючись великою компанією, «IBM» – лише одна з багатьох, але ніяк не лідер.

Звичайно нова компанія засновує свій перший продукт на якомусь технологічному нововведенні. Цей перший продукт проектується, виходячи із внутрішніх уявлень про те, як все варто робити. На цьому етапі клієнти компанії ще не проявляють великої зацікавленості, тому їхні ради нескладні. Але як тільки продукт з'являється на ринку, зацікавленість клієнтів починає збільшуватися, оскільки вони вкладають у продукт свій час і енергію. Природно, що від них приходять вимоги щодо змін і доповнень.

Компанія одна, а клієнтів у неї будуть десятки й сотні. Якщо реагувати на всіх (або хоча б на самих великих), хто візьме на себе обов'язок урегулювання цих суперечливих вимог?

Клієнт не володіє двома найважливішими якостями: 1) не піклується про інтереси розробника; 2) не знає, як проектувати.

Навіть викрикуючи суперечливі накази, клієнти очікують, що *розробник* самостійно вибере правильні рішення.

Якщо продукт розробляється під дудку покупця, він мутує від версії до версії, замість того, щоб зростати впорядковано. В остаточному підсумку продукт складається з несумісних фрагментів і випадково підібраних можливостей. Кожному клієнтові доводиться продиратися через продукт,

знаходячи можливості, які йому подобаються, уникаючи можливостей, які йому не підходять, і всі, без винятку, клієнти знаходять, що користуватися продуктом стає усе складніше й складніше з кожною новою версією.

Такий продукт не бажають купувати – розробник стає досить уразливою мішенню конкурентів.

Так як же проектувати правильно?

У процесі проектування знадобляться незаслужено забуті інструменти ручка й папір. Використання комп'ютера само по собі повільно, по-перше, оскільки інтерфейс програм недосконалий, а по-друге, через те, що, використовуючи комп'ютер, людина буде підсвідомо намагатися зробити роботу *красиво*, а не просто буде фіксувати свою думку. Відомо, наприклад, що текстові процесори не принесли прискорення письма – людина, яка пише на комп'ютері витрачає багато часу на полірування фраз. Навіщо полірувати до блиску варіант, який ви відкинете через п'ять хвилин?

Проектування складається з наступних етапів:

1. Визначення необхідної функціональності системи;
2. Створення користувацьких сценаріїв;
3. Проектування загальної структури;
4. Конструювання окремих блоків;
5. Створення глосарія;
6. Збірка й початкова перевірка повної схеми системи.

Кожний наступний етап у такій системі залежить від результатів попередніх етапів. Відповідно, пропуск якого-небудь етапу (за винятком, хіба що, створення глосарія) негативно впливає на результати всіх наступних.

На першому етапі необхідно визначити функціональність майбутньої системи. Саме функціональність буде базою для всього інтерфейсу.

Щоб правильно визначити список функцій необхідно провести *аналіз цілей* і *аналіз дій користувачів.* Для доказу цього проектувальник взаємодій Скотт Мак-Грегор на своїх заняттях використає такий чудовий тест. Він описує продукт за допомогою переліку функцій і просить слухачів записати, що це за

продукт, як тільки вони здогадаються. Він перераховує: 1) двигун внутрішнього згоряння; 2) чотири колеса з гумовими покришками; 3) трансмісія, яка зв'язує двигун з ведучими колесами; 4) трансмісія й двигун змонтовані на ходовій частині; 5) кермове колесо. На цей момент часу кожний слухач уже записав, що це автомобіль, але тут Скотт перестає описувати особливості продукту й замість цього називає пару завдань потенційного користувача: 6) швидко й легко зрізує траву; 7) на цьому зручно сидіти. На підставі п'яти функцій-підказок жоден слухач не може догадатися, що це мінітрактор-газонокосилка. Очевидно, що цілі користувача набагато більше наочні, ніж набір функцій продукту.

Аналіз цілей користувачів. Штат Детройт робив гігантські хромовані ненажерливі автомобілі й лицемірно затверджував: «Ми даємо споживачам те, що їм потрібно». Під час нафтової кризи сімдесятих японці вийшли на ринок з економічними невеликими автомобілями й нанесли Детройту удар, який не забудеться ніколи. Сьогодні автомобільна індустрія Америки проявляє набагато більшу повагу до бажань споживача й уже не насмілюється затверджувати, що все знає краще.

Японія захопила позиції на автомобільному ринку, виконавши бажання користувачів, про які ті й не підозрювали. При цьому споживачі здатні відрізнити гарну річ від поганої, якщо користувачам дадуть її побачити.

Ідеєю, яка лежить в основі даного методу, є просте міркування, яке говорить, що людям не потрібні інструменти самі по собі, потрібні лише результати їхньої роботи. Нікому не потрібний молоток, якщо не потрібно забивати цвяхи. Нікому не потрібний текстовий процесор, але потрібна можливість, зі зручністю, писати тексти. Нікому не потрібна програма обробки зображень – потрібні вже оброблені зображення.

Різницю підходів до вибору функціональності в такій системі зручно проілюструвати на прикладі тостера. Стандартний підхід, при якому функції вибираються фактично довільно, у найкращому разі приведе до такого завдання: «Потрібний ящик з вузькою прямокутною діркою й нагрівачем

усередині». Аналіз цілей користувача приведе до іншого формулювання: «Потрібний підсмажений хліб. Схоже, що найпростіше домогтися цього створенням ящика з діркою за формою шматка хліба й нагрівачем усередині. З іншого боку, схоже, що цей спосіб не єдиний».

Другий варіант при повному розвитку цього методу може привести не тільки до створення тостеру, але й також і ростеру (тобто пристрою, у якому можна присмажувати не тільки хліб). Головне ж інше. У жодному разі не можна дати обдурити себе непотрібною конкретикою, тобто описом того, яка повинна бути майбутня функціональність. Як правило, той самий результат можна досягти декількома різними способами, при цьому важливо не тільки реалізувати будь-який спосіб, але й вибрати кращий.

Результатом цього процесу повинен бути список цілей, наприклад, для тостера фінальний список цілей повинен виглядати дуже просто: «Повинен присмажувати дрібні предмети, переважно хліб».

Після того, як справжні цілі користувачів установлені (і доведено, що таких користувачів досить багато, щоб виправдати створення системи), приходить час вибирати конкретний спосіб реалізації функції, для чого використається другий метод – *аналіз дій користувачів*.

Досягнення майже всіх цілей жадає від користувачів здійснення певних дій.

Вертаючись до прикладу з тостером, можна вказати, що крім можливості присмажувати хліб він повинен вмикатися й вимикатися, більше того, він повинен бути влаштований таким чином, щоб його можна було зручно мити. З іншого боку, можливо, що тостер можна зробити так, щоб він не вимагав мийки, у цьому випадку функція «можливість мити» стає зайвою.

Суть полягає в тому, щоб угадати ті види діяльності, які буде здійснювати користувач із програмою, і сконцентруватися на тім, щоб зробити виконання діяльності простим і зручним. Розглянемо як ілюстрацію наступний приклад.

Потрібно зробити веб-сторінку, за допомогою якої можна розсилати вітальні листівки. Користуючись власною природною інтуїцією, розробник складає наступний список функцій:

1. Написати текст;
2. Додати картинку;
3. Вибрати готову листівку з бібліотеки;
4. Відправити листівку електронною поштою/на печатання.

Небажання або нездатність творців додатку надати користувачеві який-небудь інтуїтивно зрозумілий спосіб обмінятися даними із програмою може закінчитися створенням типового для «Macintosh» середини 80-х інтерфейсу: спочатку програма пропонує порожню листівку зі списком меню для набору тексту, картинок, для пошуку й завантаження листівок з бібліотеки, й для розсилання. Далі користувач змушений перегортати списки меню, намагаючись угадати всі можливі команди, і займатися синтезом цих атомарних команд для того, щоб створити картинку.

За методом *планування діяльності*, спочатку описуються ті види діяльності, які можуть зацікавити користувача. Досить поговорити з потенційними користувачами, і можна скласти такий список:

1) вітання із днем народження;
2) запрошення на вечірку;
3) вітання з ювілеєм.

Тобто з погляду користувача дії які він робить:

1) відправляє листівку «Із днем народження»;
2) планує вечірку й запрошує гостей;
3) відправляє листівку «З ювілеєм».

Очевидно, що замість того щоб почати з порожньої листівки, можна запропонувати користувачеві вибір:

Що Ви хочете зробити?

- відправити привітання із днем народження;
- відправити привітання з ювілеєм;

- відправити запрошення;
- почати з порожньої листівки.

Основні дії тягнуть за собою додаткові. Наприклад, відправляючи привітання із днем народження або ювілеєм, користувач можете захотіти, щоб йому у наступному році нагадали привітати цю людину. Тому доцільно буде додати галочку «нагадати в наступному році». Запрошення ж передбачає реакцію запрошуваного, тому розробнику може спасти на думку ідея, додати опцію збору повідомлень від запрошуваних електронним шляхом. Подібні можливості залишаються непоміченими, коли програма розробляється, виходячи з функціональності програми, а не з діяльності користувача.

2.4. Прототипування

Після того, як визначені основні концепції проекту, розробляється прототип створюваного додатку, який відображує деякі основні аспекти його функціонування.

Усе починається з розробки сценаріїв, тобто опису конкретних ситуацій, у яких можливе застосування програмного продукту. Основна помилка більшості розробників це ретельна розробка інтерфейсу перш ніж пророблені всі завдання, які інтерфейс повинен вирішувати. У таких сценаріях повинно бути відображено участь людей, обстановка, у якій вони перебувають і завдання, які вони виконують, а так само, які завдання й на якому етапі приводять до використання розроблювального додатку, що мотивує людину його використати, і як додаток дозволяє розв'язати завдання. Відходячи від конкретних елементів інтерфейсу, розробка сценаріїв дозволяє визначити всю послідовність дій для виконання завдання так, щоб найбільше логічно розмістити елементи інтерфейсу по екранних формах.

Від більш загального виду сценаріїв, які в основному зосереджені на ситуації (іноді називаються *історії використання*) необхідно перейти до сценаріїв екранних форм.

Сценарій – це стислий опис способів застосування програмного продукту персонажем для досягнення мети.

Ефективність сценарію визначається в більшій мірі його охватом, ніж глибиною. Інакше кажучи, важливіше, щоб сценарій описував процес від початку до кінця, ніж щоб він описував кожний крок у вичерпних подробицях.

Важливо розвивати лише ті сценарії, які дозволяють просуватися вперед у процесі проектування й не плутати в нетрях виняткових ситуацій. Зазвичай розробляється лише два види сценаріїв, хоча сценаріїв кожного виду може бути й декілька. Це сценарії повсякденні й сценарії обов'язкові.

Повсякденні сценарії – самі корисні й важливі. Вони описують основні дії, які користувач виконує найчастіше. Так для системи супроводу програмних помилок типові повсякденні сценарії – пошук записів про дефекти й заповнення форм про вперше виявлені дефекти. Будь-який співробітник технічної підтримки виконує ці два завдання щодня й багато разів.

У загальному випадку для більшості користувачів репертуар повсякденних сценаріїв виявляється досить обмеженим. Частіше це один або два сценарії. І рідко їх буває більше трьох.

Повсякденні сценарії мають потребу в самій надійній підтримці якісною взаємодією. Нові користувачі повинні швидко опановувати такими сценаріями, як наслідок потрібна якісна підтримка швидкого навчання. Тобто інструкції із застосування повинні бути написані у програми «на чолі». Однак частота повторень незабаром приводить до непотрібності інструкцій, тому в користувачів швидко з'являється потреба в прискорених методах роботи, таких як клавіатурні скорочення. Крім того, в міру придбання досвіду користувачі починають відчувати потребу в пристосуванні повсякденних взаємодій під індивідуальний стиль роботи й особисті переваги.

Обов'язкові сценарії описують всі дії, виконувані нечасто, але неухильно. Очищення баз даних і створення виняткових запитів можуть виявитися саме в цій категорії сценаріїв. Обов'язкова взаємодія також вимагає підтримки механізму навчання.

Однак користувач ніколи не перейде на більш високий рівень проходження таких сценаріїв. Нечасте застосування означає, що будь-який користувач погодиться з механізмами, запропонованими програмою, тому індивідуалізація не буде потрібною.

Це звільняє розробників від необхідності забезпечувати той же рівень доведення якого вимагає повсякденний сценарій. Приблизно так само розкішний інтер'єр салону нового Ягуару відрізняється від грубого металу його підкапотного простору.

У більшості продуктів репертуар обов'язкових сценаріїв досить обмежений, але кількість останніх звичайно перевищує кількість сценаріїв повсякденних.

Третій вид сценаріїв - *сценарії для виняткових ситуацій*. Програмісти природно звертають особливу увагу саме на них, але в процесі проектування інтерфейсу продукту ці сценарії можна ігнорувати. Це не означає, що відповідної функціональності немає місця в програмі, але означає, що взаємодію для таких сценаріїв можна проектувати грубо й ховати в глибини інтерфейсу. Працездатність коду може залежати від того, чи обробляються виняткові ситуації, а успіх продукту залежить від здатності справлятися з випадками, описаними в сценаріях повсякденних і обов'язкових.

Припустимо, що необхідно розробити сценарії для майбутньої поштової програми. Зважаючи на все, для цього завдання необхідно три сценарії:

а) запустити поштову програму, скачати нову пошту. Одержавши пошту, прочитати всі повідомлення, потім частину їх видалити, а на одне повідомлення відповісти. Після чого виключити поштову програму;

б) зробити активним вікно вже відкритої поштової програми й включити процес скачування нової пошти. Одержавши пошту, прочитати її. Одне повідомлення переслати іншому адресатові, після чого видалити його, а ще одне друкувати. Перемкнутися на інше завдання;

в) прийшло нове повідомлення, про що повідомляє відповідний індикатор. Зробити активним вікно поштової програми й відкрити отримане повідомлення.

Прочитати його, після чого перемістити його в іншу папку. Після чого перемкнутися на інше завдання.

Користь цих сценаріїв двояка. По-перше, вони будуть корисні для наступного тестування. По-друге, сам факт їхнього написання зазвичай (якщо не завжди) приводить до кращого розуміння улаштування проектованої системи, спонукаючи відразу ж оптимізувати майбутню взаємодію.

Справа в тому, що на таких сценаріях дуже добре помітні непотрібні кроки, наприклад, у третьому сценарії після одержання сигналу індикатора необхідно було відкрити вікно системи, знайти потрібне повідомлення, відкрити його й тільки тоді прочитати. Від цих непотрібних етапів треба позбутися вже на ранньої стадії проектування.

Наступний етап це створення *паперового прототипу*, тобто послідовних зображень функціональних блоків, намальованих від руки для виконання кожного завдання додатку.

Основні переваги паперового прототипування:

- швидкість. Накидати схематично на папері швидше, чим за допомогою спеціалізованого програмного забезпечення;
- фіксація ідеї. Важливо швидко зафіксувати ідею, а не доводити до ідеалу деталі;
- легкість зміни. Переставити елементи в намальованому за хвилину прототипі набагато легше, ніж міняти ті ж елементи в графічному редакторі з вже відрисованими тінями й відблисками;
- дешеве тестування. Паперовий прототип інтерфейсу легко протестувати показавши його майбутнім користувачам і програмістам;
- легкість створення. Для розробки прототипів інтерфейсу не потрібно вміти програмувати або малювати. Головне – розуміння принципів роботи правильного інтерфейсу;
- живе тестування. У процесі тестування є можливість легко змінювати принципи роботи інтерфейсу, відразу фіксувати ідеї, і знову давати прототип

інтерфейсу на оцінку користувачеві. Намальований на папері прототип не шкода викинути;

- прозорість інтерфейсу. При оцінці зручності використання графічні ефекти тільки відволікають;
- доступність. Навіть технічно непідковані люди можуть використати інструмент паперового прототипу. Навіть технічно непідковані люди можуть висловити свою думку про прототип інтерфейсу;
- коментування. Свої коментарі можуть залишати всі бажаючі прямо на прототипі.

Основні недоліки:

- пасує не кожній ситуації. Наприклад, неохайні ескізи на папірцях не варто показувати замовнику;
- тестування паперового прототипу не виявляє всіх проблем інтерфейсу (пов’язаних зі смугами прокручування, шрифтами, швидкістю роботи);
- користувач може працювати з паперовим прототипом зовсім інакше, ніж з реальним інтерфейсом;
- паперовий прототип може бути неточний. Якщо розробник намалював прекрасний інтерфейс – це ще не гарантує, що він вміститися на відведений йому простір на екрані;
- ефект «паперового» прототипу проявляється тільки під час обговорення.

Після тестування паперового прототипу можна переходити до створення його електронної версії. Тут уже можна більше уваги приділити деталям. Але головне завдання електронного – це проектування переходів між функціональними блоками. Діючі прототипи звичайно найбільш повно дозволяють оцінити якість механізму взаємодії користувача з розроблювальним додатком, тобто якість інтерфейсу.

Блоки можуть бути представлені як у вигляді окремої екранної форми так і бути згруповані за змістом. Важливо пам'ятати, що отримані зв'язки дуже істотно впливають на навігацію в межах системи (особливо, коли система багатовіконна). Варто уникати як занадто окремих блоків (їх важко знайти), так

і блоків, пов'язаних з більшою кількістю інших, щоб не перевантажувати інтерфейс. Для одного блоку оптимально мати зв'язок з трьома іншими блоками.

Навігаційна схема допомагає побачити кількість переходів, які треба зробити користувачеві в рамках одного сценарію. Усунути проблеми пов'язані з розміщенням функцій одного блоку в різних екранах, забрати зайві екрани й оптимально скомпонувати функції.

При розробці прототипів інтерфейсу для кожного блоку до візуальних атрибутів відображуваної інформації відносяться:

- взаємне розташування й розмір відображуваних об'єктів;
- колірна палітра;
- засоби залучення уваги користувача.

Проектування розміщення даних на екрані припускає виконання наступних дій:

1) визначення складу інформації, яка повинна з'являтися на екрані;

2) вибір формату подання цієї інформації;

3) визначення взаємного розташування даних (або об'єктів) на екрані;

4) вибір засобів залучення уваги користувача;

5) розробка макета розміщення даних на екрані;

6) оцінка ефективності розміщення інформації.

Процес проектування повторюється доти, поки розроблювач і потенційні користувачі не будуть задоволені.

Загальні принципи розташування інформації на екрані повинні забезпечувати для користувача:

- можливість перегляду екрану в логічній послідовності;
- простоту вибору потрібної інформації;
- можливість ідентифікації зв'язаних груп інформації;
- відмінність виняткових ситуацій (повідомлень про помилки або попереджень);

- можливість визначити, яка дія з боку користувача потрібна зараз (чи потрібна взагалі) для продовження виконання завдання.

Прототипи за своєю природою являють собою програми, створювані в поспіху з метою перевірки деяких припущень. Щоб швидко створити прототип, програміст повинен пожертвувати ідеальним вирівнюванням цеглин. Тут не використаються «правильні» структури даних, «правильні» алгоритми, інформація безсистемна, використовуються будь-які фрагменти коду, що «підходять». Тому такий код може бути застосовано тільки на даному етапі.

2.5. Випробування програмного продукту

Спільне з потенційним користувачем випробування створюваного програмного продукту забезпечує одержання достатньо цінної додаткової інформації і є, як правило, підставою наступної успішної реалізації продукту. Необхідно підкреслити, що випробування програмного продукту принципово відрізняється від його налагодження.

Перша й найбільш важлива відмінність обумовлена різними цілями цих двох процесів: налагодження має метою виявлення дефектів (помилок) програмування, у той час як у ході випробувань оцінюється, наскільки повно розроблений додаток (зокрема, його інтерфейс) відповідає потребам і очікуванням користувача. Звичайно, наявні програмні помилки можуть іноді вплинути на якість інтерфейсу, але це скоріше виключення, чим правило.

Друга принципова відмінність полягає в тому, що налагодження виконує безпосередньо його розробник, а основною діючою особою при проведенні випробувань є потенційний користувач (замовник). Завдяки цьому в ході випробувань можуть бути виявлені не тільки технічні помилки, але й концептуальні проблеми в запропонованому продукті.

Крім того, випробування можуть проводитися для двох або більше альтернативних варіантів реалізації створюваного додатку з метою виявлення найбільш удалого саме з погляду користувача й розв'язуваних ним завдань.

Як результат випробувань велике значення мають не тільки кількісні (об'єктивні) дані про роботу додатку в тих або інших ситуаціях, але й «якісна» інформація, яка відображує суб'єктивне сприйняття користувачем запропонованого варіанту додатку, його задоволеність, а також перелік проблем, які на його погляд можуть мати місце при реальній експлуатації програмного продукту.

Як було зазначено вище, основна мета випробувань – визначити, наскільки повно розроблений додаток (у першу чергу, його інтерфейс) відповідає потребам і очікуванням користувача. У зв'язку з цим основним напрямком випробувань додатку є *оцінка його юзабилити (usability) – «споживчих властивостей».* Така оцінка повинна проводитися, починаючи із самих ранніх етапів розробки. Основою для проведення оцінки повинні служити дані про те, як користувачі зазвичай виконують ту роботу, яку покликаний автоматизувати створюваний додаток. У міру послідовного виконання розробки, оцінка споживчих властивостей додатку повинна постійно уточнюватися. Чим частіше й коректніше буде проводитися оцінка, тим вище буде якість розробки.

Якість будь-якого інтерфейсу в остаточному підсумку визначається якістю взаємодії між однією людиною й однією системою. *Якщо індивідуальна взаємодія з деякою системою не проходить для користувача легко й комфортно, то в результаті цей недолік негативним образом відображується на якості роботи всієї системи, незалежно від того, наскільки вона хороша в інших своїх проявах.*

Повторне виконання етапів розробки. Оскільки випробування часто виявляють ті або інші слабості проекту, або, принаймні, забезпечують одержання додаткової інформації, яку розробник захоче використати, майже завжди виявляється необхідним повернення до одного з попередніх етапів розробки (а іноді й у початкову точку) і проведення повторних випробувань. Так може тривати до тих пір, поки й розробник, і потенційні користувачі не будуть повністю задоволені отриманими результатами.

Тема 3. Властивості людино-машинного інтерфейсу

Для створення в користувача відчуття «внутрішньої волі» інтерфейс повинен володіти цілим рядом властивостей, розглянутих нижче.

Природність інтерфейсу (інтуїтивно зрозумілий інтерфейс). Природний інтерфейс – такий, що не змушує користувача істотно змінювати звичні для нього способи розв’язання завдання. Це означає, що повідомлення й результати, які видає відповідний програмний продукт, не повинні вимагати додаткових пояснень. Доцільно також зберегти систему позначень і термінологію, які використовуються в даній предметній області.

Використання знайомих користувачеві понять і образів (метафор) забезпечує інтуїтивно зрозумілий інтерфейс при виконанні завдань. Разом з тим, застосовуючи метафори, не треба обмежувати їхню машинну реалізацію повною аналогією з однойменними об'єктами реального миру. Наприклад, папка на Робочому столі Windows може бути використана для зберігання цілого ряду інших об'єктів, ніж звичайна папка. Метафори є свого роду «містком», який зв'язує образи реального миру з тими діями й об'єктами, якими доводиться маніпулювати користувачеві при його роботі на комп'ютері; метафори забезпечують «впізнавання», а не «згадування». Користувачі запам'ятовують дію, зв'язану зі знайомим об'єктом, легше, ніж вони запам'ятовували би ім'я команди, пов'язаної із цією дією.

Погодженість інтерфейсу. Погодженість дозволяє користувачам переносити знання, які в них вже є, на нові завдання, освоювати нові аспекти швидше, і завдяки цьому фокусувати увагу на завданні, яке розв’язується, а не витрачати час на з'ясування розходжень у використанні тих або інших елементів управління, команд і т. п. Забезпечуючи *спадкоємність* отриманих раніше знань і навичок, погодженість робить інтерфейс *розпізнаваним і передбачуваним*.

Погодженість важлива для всіх аспектів інтерфейсу, включаючи імена команд, візуальне подання інформації й поведінку інтерактивних елементів.

Для реалізації погодженості в створюваному програмному забезпеченні, необхідно враховувати різні аспекти.

Погодженість у межах продукту. Та сама команда повинна виконувати ті самі функції, де б вона не зустрілася, при чому тим самим чином. Наприклад, якщо в одному діалоговому вікні команда Копіювати означає негайне виконання відповідних дій, то в іншому вікні ця команда не повинна вимагати від користувача додатково вказувати розташування інформації, яка копіюється. Треба використати ту саму команду, щоб виконати функції, які користувачу здаються подібними.

Погодженість у межах робочого середовища. Підтримуючи погодженість із інтерфейсом, який надається операційною системою, додаток може «спиратися» на ті знання й навички користувача, котрі останній одержав раніше при роботі з іншими додатками.

Розробник може влаштувати, у тому числі й ненавмисно, пастку для користувача, якщо зробить так, що на одному комп'ютері будуть інтенсивно використовуватися два або більше додатки, інтерфейси яких відрізняються тільки декількома деталями, які часто застосовуються. У таких обставинах у користувача, швидше за все, сформуються звички, які будуть приводити до помилок при спробах застосувати в одному додатку команди, притаманні іншому.

Погодженість у використанні метафор. Коли ми переносимо знання про навколишнє середовище у світ комп'ютерів, починає діяти концепція метафор. Наприклад в Windows метафорами є: робочий стіл, папки, кошик і т.д.

Якщо поводження деякого програмного об'єкта виходить за рамки того, що звичайно мається на увазі під відповідною йому метафорою, у користувача можуть виникнути труднощі при роботі з таким об'єктом. Наприклад, якщо для програмного об'єкта Кошик визначити операцію Запуск, то для з'ясування її змісту користувачеві, швидше за все, буде потрібна стороння допомога.

Якщо вже є вдала метафора, яка використовується в додатку, яким вже користуються мільйони людей, наприклад Word, Excel, або Internet Explorer, то

вони будуть думати, що така ж метафора точно так само буде працювати й в іншому додатку. Ідентична ситуація з інтернет-додатками. В інтернет-магазині на amazon.com кошик з покупками покупця зберігається протягом 90 днів. Тобто є користувачі, які звикли до того, що залишивши товар у кошику, його можна знайти там через 2 тижні. Якщо раптом у новому інтернет-магазині кошик буде очищатися через 24 години, це стане для користувачів великим сюрпризом, і навряд чи вони захочуть повторити все заново.

Ще одним стандартом є вікна діалогу. Наприклад, у діалогах підтвердження кнопка Так завжди перебуває лівіше за кнопку Ні. Якщо поміняти їх місцями величезна кількість користувачів по інерції будуть натискати не ту кнопку й тим самим можуть видалити багато корисних файлів або нанести іншої непоправної шкоди. Змінити зовнішній вигляд вікна діалогу можна, але не за рахунок зміни функціональності.

Дружність інтерфейсу (принцип «пробачення» користувача). Користувачі звичайно вивчають особливості роботи з новим програмним продуктом методом проб і помилок. Ефективний інтерфейс повинен брати до уваги такий підхід. На кожному етапі роботи він повинен дозволяти тільки відповідний набір дій і попереджати користувачів про ті ситуації, де вони можуть ушкодити системі або даним; ще краще, якщо в користувача існує можливість скасувати або виправити виконані дії.

Приклади дружності:

1. Розробники радіоприймачів і телевізорів із цифровими настройками завжди передбачають можливість збереження параметрів звуку й каналів, на які ці прилади були настроєні востаннє.

Повернення користувача після завантаження програми до того місця, де він зупинився, дозволяє йому швидше включитися в роботу. Зручно, коли не доводиться перегортати книгу спочатку, згадуючи на якій сторінці зупинився.

2. У випадку, коли користувач у наступний момент може виконати тільки одну дію, добре, якщо цю дію виконує комп'ютер.

3. У багатьох випадках користувач робить дії, які сприймаються програмою як неправильні, не тому, що він дурень, а тому, що система не показала йому границь допустимої дії.

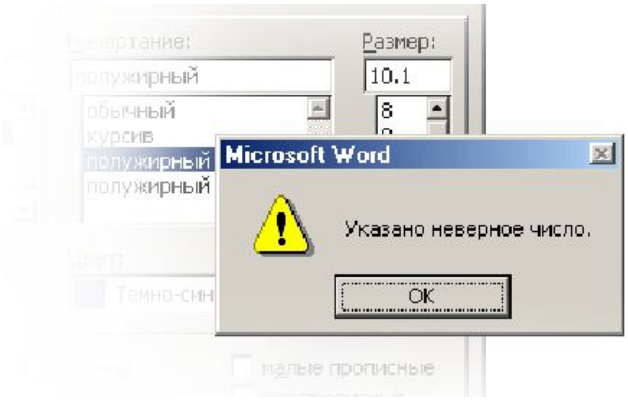


Рис. 3.1 Типове повідомлення про помилку

Повідомлення про помилку на рис. 3.1 викликає в користувача масу питань. Чому число не вірне? Хіба це не число? Шрифт не може бути більше 10? Судячи із запропонованої шкали, може.

Усіх цих питань можна було б уникнути, якщо написати, що розмір шрифту винний бути кратним 0,5. Наприклад, 10.0 або 10.5.

Рекомендації щодо написання повідомлення про помилку:

- розповісти користувачеві, що він зробив;
- пояснити проблему;
- пояснити, як виправити помилку;
- використовувати активний, а не пасивний стан;
- використовувати тільки позитивний відтінок (наприклад, замість «не пишіть кирилицею» написати «пишіть тільки латинськими буквами»);
- складати повідомлення без образ і спроб пожартувати;
- ураховувати культурні особливості країни, де продається програмний продукт. Повідомлення доречне в культурі однієї країни може бути неприпустимим в іншій;
- навести приклад.

Ще одне невдале повідомлення про помилку: «До того, як рахунок може бути оплачений, необхідно, щоб дата платежу була пізніше, ніж дата створення рахунку».

Відповідно до рекомендацій наданих вище, повідомлення повинне виглядати так: «Ви ввели дату платежу ранішу за дату створення рахунку. Змініть дату таким чином, щоб дата платежу була пізніше дати створення рахунку».

Краще повідомлення про помилку – це відсутність повідомлення про помилку. У випадку з «невірним числом» можна просто ігнорувати неправильне значення, округляючи його до найближчого, яке дозволяється (можливо, звернувши увагу користувача на самостійність дій системи однократним миготінням відповідного поля уведення). Або замість звичайного поля уведення можна використати крутилку (див. Тема 6).

Принцип «зворотного зв'язку». Кожна дія користувача повинна одержувати *візуальне*, а іноді й *звукове підтвердження* того, що програмне забезпечення сприйняло уведену команду; при цьому вид реакції, по можливості, повинен ураховувати природу виконаної дії.

Ніщо так не бентежить не дуже досвідченого користувача, як заблокований екран, який не реагує на його дії. Типовий користувач здатний витерпіти лише кілька секунд очікування відповідної реакції від свого електронного «співрозмовника».

Кожний об'єкт, над яким можна зробити дію, повинен це показувати всім своїм видом, а заодно й спосіб дій користувача, на який цей об’єкт реагує (рис. 3.2).

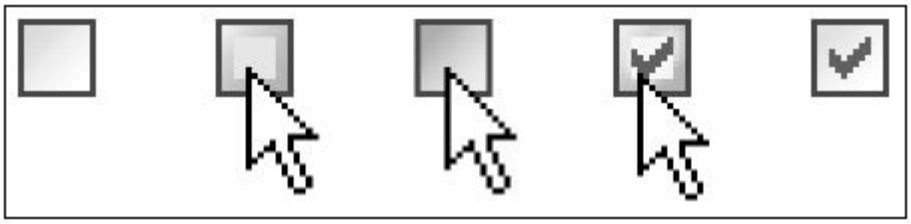


Рис. 3.2 Візуальна зміна чек-боксу при різному положенні курсору миші, що є постійним зворотним зв'язком з користувачем.

Гнучкість інтерфейсу — це його здатність ураховувати рівень підготовки й продуктивність праці користувача, його індивідуальні особливості.

Гнучкість припускає можливість зміни структури діалогу й/або вхідних даних. Концепція *гнучкого* ***(адаптивного)*** інтерфейсу на сьогодні є однією з основних областей дослідження взаємодії людини й ЕОМ.

Перший аспект гнучкого інтерфейсу це *рівень підготовки користувача*. Чи потрібно налаштовувати інтерфейс під рівень користувача і як визначати цей рівень?

Користувач складної системи не є ні новачком, ні експертом. Тому що він може знати або не знати кожний елемент інтерфейсу, або кожний набір зв'язаних елементів, які працюють однаковим образом. Можна знати, як використати більшість команд і опцій якогось програмного пакета й навіть працювати з ним професійно. При цьому можна не знати, як користуватися деякими іншими командами, або взагалі не знати про існування цих команд у даному пакеті. Наприклад, користувачеві програми для обробки фотозображень, який застосовує її тільки для створення онлайнових зображень, може ніколи не знадобитися можливість виконання кольороподілу, яка використовується, головним чином, для комерційного друку.

Розробники інтерфейсів робили різні спроби врахувати припущення про те, що користувачів можна розділити на новачків і експертів. Але оскільки це припущення невірно, всі ці спроби, природно, провалилися. Добрим прикладом можуть послужити адаптивні системи, здатні автоматично перемикатися з режиму для починаючих користувачів у режим для досвідчених користувачів, коли вони визначають, що ваше вміння володіти системою досягло необхідного рівня. Якщо під час використання цієї системи в режимі для починаючих користувачів вона раптово перемикається в «експертний режим», для користувача відбувається несподівана зміна робочого середовища, принаймні, його частини. Не кращим варіантом є й те, якщо система буде перемикатися поступово, частина за частинами. У цьому випадку вона буде проявляти себе

хитливо й безладно, оскільки звички й навички, які вчора сформувалися, стають марними, якщо сьогодні даний елемент перемкнувся в експертний режим.

Другий аспект гнучкого інтерфейсу це *персоналізація*. Установка персональних настроювань у спільно використовуваному середовищі здатна викликати серйозні збої, тому що це означає, що в інтерфейс можуть бути непомітно внесені зміни. У результаті вчора правильною дією було натискання на червону кнопку, а сьогодні – на синю, тому що комусь здалося, що сині кольори красивіші.

Третій аспект – гнучкість у роботі, властива людям. Наприклад, цифрова система, перш ніж видати накладну, вимагає надати інформацію про покупців і інформацію про замовлення. Клерк може просто помістити замовлення на обробку, ще не маючи повної інформації про покупця, а ось комп’ютеризована система відхилить транзакцію, не бажаючи продовжувати роботу без необхідних відомостей.

Гнучкість, яка дозволяє виконувати дії до задоволення попередніх умов, звичайно стає однією з перших жертв комп'ютеризації, а її відсутність – головна причина нелюдськості цифрових систем. Це природний результат застосування моделі реалізації. Програмісти не бачать причин для створення проміжних станів, оскільки комп'ютер у них не має потреби. Однак людина має потребу в можливості трішки змінювати систему.

Простота інтерфейсу. Інтерфейс повинен бути простим. При цьому мається на увазі не спрощенство, а забезпечення *легкості* в його *вивченні* й у *використанні*.

Крім того, інтерфейс повинен надавати доступ до всього переліку функціональних можливостей, передбачених даним додатком. Реалізація доступу до широких функціональних можливостей і забезпечення простоти роботи суперечать один одному. Розробка ефективного інтерфейсу покликана збалансувати ці цілі.

Один з можливих шляхів підтримки простоти — надання допомоги користувачам у управлінні складністю відображуваної інформації, використовуючи *послідовне розкриття* (діалогових вікон, розділів меню й т. п.). Послідовне розкриття припускає таку організацію інформації, при якій у кожний момент часу на екрані перебуває тільки та її частина, яка необхідна для виконання чергового кроку. Скорочуючи обсяг інформації, представленої користувачеві, зменшують обсяг інформації, яка підлягає обробці.

Інший шлях до створення простого, але ефективного інтерфейсу — *розміщення* й подання елементів на екрані з обліком їхнього *змістовного значення* й логічного взаємозв'язку. Це дозволяє використати в процесі роботи асоціативне мислення користувача.

Аспекти інтерфейсу, які вимагають спрощення, — тексти. Користувачі працюють над рішенням будь-якого завдання, і для них читання довідки уявляється лише непотрібними витратами часу, або ж перешкодою, яка відволікає від досягнення мети. Користувачі читають керівництва винятково за принципом «до зарізу треба зробити, але нема кого запитати як».

Тому необхідно проектувати програму так, щоб потреба в читанні керівництва не виникала. Єдине виключення, коли в користувачів немає базових знань у прикладній області, але вони почувають, що це знання їм необхідно. Доброю ілюстрацією цього виключення є програма для бухгалтерського обліку в малому бізнесі QuickBooks від «Intuit». Більшість із тих, хто цією програмою користується – представники малого бізнесу, які не мають ніякого уявлення про те, що саме є бухгалтерський облік. Автори керівництва до QuickBooks знали про це, як і про те, що їм потрібно навчити людей основним принципам бухгалтерського обліку. Без керівництва користуватися QuickBooks було б неможливо. З іншого боку, люди, знайомі з бухгалтерським обліком, можуть спокійно працювати із програмою без керівництва.

Користувачі не читають не тільки керівництва, але й повідомлення-підказки.

При проведенні *юзабіліті-тестування* подібних речей можна виявити, що:

• просунуті користувачі подібні інструкції пропускають. Вони знають, що треба робити, і часу на читання складних інструкцій не мають;

• більшість недосвідчених користувачів подібні інструкції пропускають. Вони не люблять читати й розраховують на те, що установки за замовчуванням їм підійдуть;

• недосвідчені користувачі, які залишилися, чесно намагаються прочитати інструкції (деякі з них просто тому, що беруть участь у тестуванні й вважають себе зобов'язаними виконувати всі інструкції). Але їх часто приводить в утруднення сама кількість слів і понять. Навіть якщо з першою появою діалогового вікна вони були достатньо упевнені у власних уявленнях дії, які необхідно зробити, багатослівні інструкції приводять користувачів у скрутний стан.

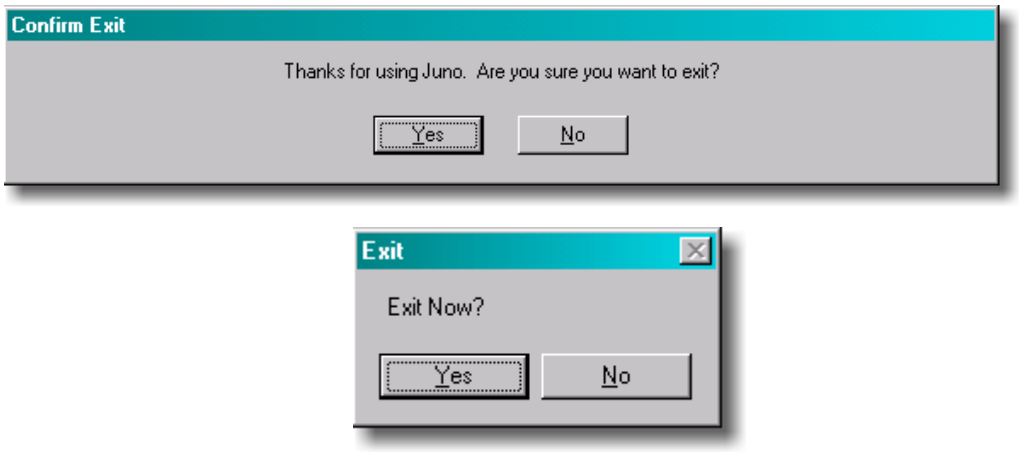


Рис. 3.3 Повідомлення, які вимагають дозволу на вихід із програми

Тобто із запропонованих на рис. 3.3 діалогових вікон друге буде більш переважним. Відразу задається питання по суті й два слова швидше за все будуть прочитані. До того ж вони зрозумілі навіть людині, яка не знає англійської мови.

Багато програм обходяться взагалі без подібного дозволу. Вони просто зберігають версію вашої роботи до змін і після й при наступному зверненні пропонують вибір.

Рекомендація щодо створення текстових повідомлень – залиште зайві слова.

Скорочення текстових повідомлень у веб-додатках:

- знижує рівень шуму на сторінках;
- підкреслює той зміст сторінок, що дійсно є цінним і корисним;
- дозволяє скоротити сторінки так, щоб користувачі могли бачити кожну з них одним поглядом, без прокручування екрану (звичайно, якщо мова йде про електронний журнал, то на статті дана рекомендація не поширюється).

Читання й розуміння – не одне й те саме.

Для біолога не важко буде розібратися в наступному тексті:

Отриманий в результаті окисного декарбоксилювання пірувата в мітохондріях ацетил Коа вступає в цикл Кребса.

Для людини іншої професії розібратися практично неможливо. Так само, текст зрозумілий програмістові не буде зрозумілий біологові. Тому всі інструкції, які включають у програмне забезпечення, потрібно перевіряти на зручність читання.

Для *оцінки зручності читання* використається *формула легкості читання Флешу* (1). За допомогою цієї формули оцінюється не тільки легкість читання тексту, але й рівень освіти читаючого. Чим вище значення, тим легше прочитати текст і тим більшому числу читачів він буде зрозумілий.

$$205.835 - 1.015\left(\frac{слов\_в\_документе}{предложений\_в\_документе}\right) - 84.6\left(\frac{слогов\_в\_документе}{слов\_в\_документе}\right) \quad (1)$$

Можна також скористатися для більшої точності формулою-виправленням для слов'янських мов, які застосовують кирилицю (2). Адже за твердженням філологів у таких мовах середня довжина речення менше, а слова в середньому довше за романо-германські мови, які використовують латиницю.

$$206.835 - 1.3\left(\frac{слов\_в\_документе}{предложений\_в\_документе}\right) - 60.1\left(\frac{слогов\_в\_документе}{слов\_в\_документе}\right) \quad (2)$$

У цій перевірці Флеша текст оцінюється за 100-бальною шкалою. Чим вище значення, тим простіше зрозуміти документ. Для більшості стандартних документів варто домагатися значення від 60 до 70 балів.

Аспекти інтерфейсу, які вимагають спрощення, — дизайн. Дійсно мінімалістичний дизайн – дизайн із максимальним співвідношенням сигнал/шум.

Рис. 3.4 Макети таблиці з різним відношенням сигнал/шум

Сигнал — це інформація, яку потрібно повідомити; *шум* — стороння інформація, яка розмиває сигнал (рис. 3.4). Ключовим моментом у створенні мінімалістичного дизайну, є досягнення максимального сигналу й мінімального шуму. У мінімалістичному дизайні, видалення непотрібних елементів наголошує на важливій інформації, у результаті чого підсилюється сигнал і зменшується шум.

Естетична привабливість. Коли потрібний мінімалізм у дизайні? Це залежить від того, як визначається успіх дизайну. На рис. 3.5 представлено 4 макети однієї й тієї ж панелі. Якщо хороший дизайн той, котрий дуже красивий, то найкраще підходить перша панель навігації. Якщо потрібно одночасно досягти привабливого зовнішнього вигляду й не напружувати користувачів, то друга й третя панелі навігації будуть придатним вибором.

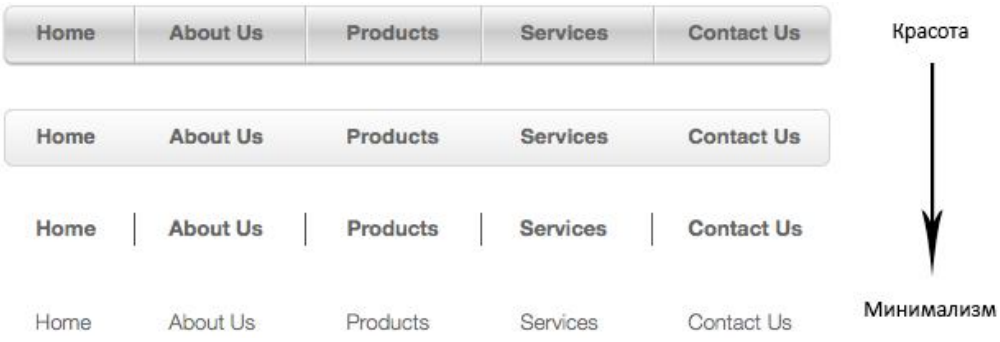


Рис. 3.5 Макети панелі з різним відношенням краса/мінімалізм

Але якщо потрібен абсолютний мінімалізм, то саме нижня навігаційна панель буде кращим вибором. Головне переконатися, що сигнал більший за шум. Мінімалізм – це добре, але іноді досить простоти.

Узагальнюючи викладене вище, можна коротко сформулювати ті основні правила, дотримання яких дозволяє розраховувати на створення ефективного користувацького інтерфейсу:

- проектувати й розробляти інтерфейс користувача як окремий компонент створюваного програмного продукту;
- враховувати можливості й особливості апаратно-програмних засобів, на базі яких реалізується інтерфейс;
- доцільно враховувати особливості й традиції тієї предметної області, до якої відноситься створюваний програмний продукт.

Тема 4. **Засоби діалогу.**

4.1.Типи діалогів

1. *Діалог типу «питання – відповідь» (*Q&A*).* У кожній точці діалогу система виводить як підказку одне питання, на яке користувач дає одну відповідь. Залежно від отриманої відповіді система може вирішити, котре питання задавати наступним.

Структура Q&A надає природний механізм уведення як управляючих повідомлень (команд), так і даних. Ніяких обмежень на діапазон або тип вхідних даних, які можуть оброблятися, не накладається.

Дана структура має ряд істотних недоліків. По-перше, не гарантує мінімального обсягу введення, оцінюваного за кількістю натискань клавіш; по-друге, можливі проблеми з аналізом й інтерпретацією даних, які вводяться; по-третє, процедура введення відповідей набором їх із клавіатури досить стомлююча для користувача. Тому в чистому виді така схема практично не застосовується.

2. *Діалог на основі меню.* Структура типу меню є найбільш природним механізмом для роботи із пристроями вказування та вибору. Це дуже популярна схема. Вона зручна користувачам, тому що їм не треба запам'ятовувати й друкувати команди й дані. Схема зручна розробникам, тому що дозволяє контролювати тип інформації, яка вводиться користувачем. Меню має безліч варіантів подання:

2.1 Список об'єктів, які обираються прямою вказівкою, або вказівкою номеру (мнемонічного коду), наприклад панель дій.

Панель дій – це надзвичайні меню, які не треба викликати. Вони завжди видимі на основному інтерфейсі. Панелі дій відмінно підходять для представлення панелей інструментів, коли функції краще пояснити вербально, ніж візуально.

2.2 Меню у вигляді блоку даних, наприклад спливаюче меню або меню, що випадає.

Спливаюче (контекстне) меню – з'являється при натисканні на праву кнопку миші або інших подібних діях на панелі або елементі. Це меню надає короткий список дій специфічних для даного контексту, а не для всього додатку.

Меню, що випадає – викликається натисканням на елементах управління інтерфейсу типу комбобокс. Однак, випадаючи елементи керування призначені для здійснення вибору на формі, а не для подання функціональності. Слід уникати його використання у таких цілях.

В кінці назви команд у таких меню ставиться три крапки, щоб позначити, що програмі буде необхідно відкрити вікно для запиту додаткової інформації, перш ніж вона зможе виконати команду.

2.3 Меню у вигляді рядка даних, наприклад лінійні меню.

Лінійні меню є стандартом для більшості десктопних додатків (додатків робочого стола). Вони показують повний набір функцій додатку, організованих найбільш передбачуваним способом (Файл, Редагування, Вид). Деякі функції діють на весь додаток, деякі тільки на обраний елемент. Лінійне меню завжди дублює функціональність контекстного меню й панелі інструментів, тому функції лінійного меню завжди доступні – видимі при перегляді екрану, і досяжні через клавіатуру. Лінійні меню з'являються в деяких веб-додатках, особливо у програмах, які емулюють десктопні аналоги (наприклад, графічні редактори).

2.4 Меню у вигляді піктограм, наприклад панель інструментів.

Панель інструментів. Канонічна панель інструментів тонкий довгий ряд кнопочок з іконками. Часто на них розташовані інші типи кнопок або елементів управління, таких як поля, або списки, що випадають. Іконкова панель інструментів працює краще, коли зображені на них дії мають явне візуальне подання. Якщо дія вимагає пояснення словами, краще використовувати інший елемент керування, такий як комбобокс або кнопки з текстовими мітками. Іконки з малюнками класичне джерело конфліктів і поганої зручності використання.

У дизайні необхідно використовувати ізоморфні іконки – тобто піктограми, які взяті з реального життя (наприклад, відро для сміття, пензлик, ластик і т. п).

3. *Діалог на основі екранної форми.* На практиці форми використовуються там, де облік будь-якої діяльності вимагає введення стандартного набору даних. Людина працює з формою до тих пір, поки не заповнить всі необхідні поля й не передасть системі. Система може перевіряти кожну відповідь безпосередньо

при введенні або по закінченні заповнення всієї форми. Перший варіант є переважним, бо без затримки реагує на дії користувача.

Структура діалогу на основі екранної форми забезпечує високий рівень підтримки: для кожної форми можуть бути передбачені повідомлення про помилки й довідкова інформація. Користувачеві можна також надати допомогу, включивши деякі елементи формату відповіді в питання або в поле відповіді.

Рекомендації щодо оформлення текстових полів:

1) довжина поля винна відповідати максимально можливій довжині його змісту. Розмір поля є інтуїтивною підказкою для користувача. Наприклад є поле для індексу. Індекс зазвичай складається з 5 цифр. Таким чином поле введення індексу не має бути розтягнуто на всю ширину форми або бути дуже маленьким. Воно повинно дозволяти користувачу ввести рівно 5 символів і всі ці символи має бути видно.

Якщо довжина інформації не є однаковою для всіх значень, то варто орієнтуватися на найдовше значення. Наприклад, для перевірки достатньої кількості символів у поле Місто можна ввести назву Белгород-Дністровський. Так само й при виділенні довжини для поля Прізвище варто враховувати подвійні варіанти.

2) не варто робити всі поля однієї довжини заради зовнішнього вигляду.

За полями введення однакової довжини губиться їх призначення. Рівні цеглини таких полів тільки виглядають охайно, проте дуже незручні у роботі. А у випадку коли форма довга, поля різної довжини виконують ще й навігаційну функцію – користувачу легше розібратися де яке поле не вчитуючись у назви.

3) треба пам’ятати про підказки та помилки.

Треба передбачати такі питання користувача: в якому форматі написати телефон, як має виглядати дата і т. п. У таких випадках потрібно робити підказки. Види підказок різноманітні й дозволяють знаходити оптимальні варіанти в залежності від ситуації. Підказки можна розміщувати за полем, під ним і навіть в середині поля. Підказка всередині поля зазвичай зникає, як тільки

воно стає активним й користувачу потрібно згадувати її зміст або навіть стирати введений текст з метою подивитися підказку.

4) потрібно показати користувачу помилки. Не дивлячись на всі підказки, він може не заповнити обов'язкове поле, ввести дані невірно.

По-перше необхідно передбачити місце для виведення помилок. По-друге продумати спосіб їх виведення. Треба чітко показати користувачеві місце, де він зробив помилку Важлива локалізація. Повідомлення про помилку повинно виводитися біля того місця, де відбулася помилка.

Існує поняття "*спливаючих вікон",* які дозволяють розширити діалог користувача з додатком. Взагалі спливаючі вікна використовуються для передачі повідомлення або підказки.

Діалогове вікно не найкращий спосіб показувати повідомлення про помилки. У Windows є елемент управління, значно краще призначений для показу повідомлень. Називається цей елемент міхур (рис. 4.1).

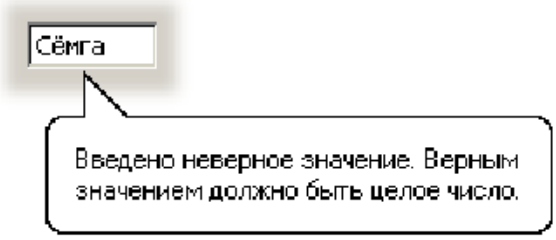


Рис. 4.1 Міхур – елемент управління для виводу підказок

Міхур, у порівнянні з діалоговим вікном, має істотні переваги. По-перше, він набагато слабкіше збиває фокус уваги, ніж вікно. По-друге, щоб закрити міхур, користувачам не обов'язково натискати на яку-небудь кнопку, досить клацнути мишею в будь-якому місці екрану. По-третє, він не перекриває значиму область системи. По-четверте, що саме головне, він показує, у якому саме елементі управління була допущена помилка.

На жаль, міхури мають і недоліки. По-перше, у них не може бути ніяких функціональних елементів. В Windows міхури взагалі реалізовані досить бідно. По-друге, міхурів взагалі немає в Інтернеті.

5) необхідно впевнитися, що назви полів добре читаються.

Назви полів можуть бути вирівняні за лівим або правими боком, знаходитися над полями. Головне – зручність читання. Дивлячи на назву, користувач завжди повинен розуміти, до якого поля вона відноситься.

6) При створені форм треба пам’ятати, що форма має бути якомога коротша й не виходити за границю екрану. Розташування назв зверху над полем робить форму довше, що не є зручним.

Користувачам не подобається заповнювати об’ємні форми, які не мають границь. Якщо немає можливості скоротити кількість інформації, яку потрібно отримати від користувача, варто розбити форму по крокам.

За можливістю треба виконувати за користувача частину його роботи. Для відомих заздалегідь варіантів даних корисно запропонувати вибір через готові списки замість простих текстових полів (детальніше про функціональні елементи див. у темі 6).

Другий варіант полегшення життя користувачу це застосування попереднього заповнення. Наприклад для полів «Країна» й «Місто» бажано застосувати автоматичне визначення місце розташування за IP комп’ютеру у мережі Інтернет. Тоді користувачеві тільки залишиться перевірити, що все вірно.

4. *Діалог на основі командної мови.* Інтерфейси командного рядка звичайно дозволяють вільну форму доступу до функціональності системи. Але функціональність такого типу «невидима», тому що більшість таких інтерфейсів нелегко розкривають доступні команди. Тобто вони сильні тільки в тому випадку, якщо користувачі вивчили все, що доступно, тоді будь-яка дія може бути зроблена за допомогою одиночної, правильно сконструйованої команди. Такі інтерфейси добрі для людей прихильних до вивчення програмування. Тому що по суті командна мова й є окремою мовою програмування, яка дозволяє дуже гнучко керувати можливостями системи. Проте коло користувачів, кому це дійсно потрібно, є дуже обмеженим.

4.2. Розробка сценарію діалогу

Сценарій діалогу – детальний опис структури й змісту діалогу, достатній для його реалізації автоматичним шляхом.

Цілі розробки сценарію:

1. Виявлення й усунення можливих «тупикових» ситуацій;
2. Вибір раціональних шляхів переходу з одного стану в інший (з поточних у потрібні);
3. Виявлення неоднозначних ситуацій, які вимагають надання додаткової допомоги користувачеві.

Складність розробки сценарію визначається в основному двома факторами – функціональними можливостями створюваного додатку (тобто числом і складністю реалізованих функцій обробки інформації) і ступенем невизначеності можливих дій користувача.

Ступінь невизначеності дій користувача залежить від обраної структури діалогу. Найбільшу детермінованість має діалог на основі меню, найменшу – діалог типу «питання-відповідь», керований користувачем.

Таким чином, сценарій діалогу можна спростити, знизивши ступінь невизначеності дій користувача. Можливі способи розв’язання цієї задачі:

- використання змішаної структури діалогу (застосування меню з метою «обмеження волі» користувача там, де це можливо);
- застосування вхідного контролю інформації, яка вводиться (команд і даних).

Ухвалюючи рішення щодо подання даних потрібно враховувати ряд факторів:

1) що потрібно користувачеві: точні значення даних або співвідношення між значеннями?

2) наскільки швидко будуть відбуватися зміни даних? Чи потрібно негайно показувати користувачеві зміну значень?

3) чи винен користувач застосовувати будь-які дії у відповідь на зміну даних?

4) чи потрібно користувачеві взаємодіяти з відображуваною інформацією за допомогою інтерфейсу із прямим маніпулюванням?

5) інформація повинна відображатися в текстовому (описовому) або числовому форматі? Чи важливі відносні значення елементів даних?

Додаткові можливості зі зниження невизначеності дій користувача надає об’єктно-орієнтований підхід до розробки інтерфейсу, при якому для кожного об'єкта заздалегідь установлюється перелік властивостей і припустимих операцій.

Найбільш ефективний такий підхід при створенні графічного інтерфейсу. Скорочуючи число можливих станів діалогу, розробник повинен пам'ятати про необхідність відображення в його сценарії роботи засобів підтримки користувача (що ускладнює сценарій).

Режими *(modes)* є важливим джерелом помилок, плутанини, непотрібних обмежень і складності в інтерфейсі. Багато проблем, до яких приводять режими, добре відомі. Проте, практика створення систем без режимів чомусь не знаходить широкого застосування в розробці інтерфейсів.

Щоб розуміти режими, варто спочатку визначити поняття жесту. *Жест (gesture)* — це послідовність дій людини, яка виконується автоматично (після старту).

Наприклад, для досвідченого користувача набір такого простого слова, як *воно*, представляє всього один жест, у той час як для починаючого користувача, який ще не цілком опанував клавіатуру, набір цього слова по буквах буде складатися з окремих жестів.

У більшості інтерфейсів допускається кілька інтерпретацій конкретного жесту. Режими проявляються в тому, як реагує інтерфейс на той або інший жест. Стан інтерфейсу, при якому інтерпретація даного конкретного жесту залишається незмінною, називається *режимом.* Якщо жест одержує іншу інтерпретацію, це означає, що інтерфейс перебуває в іншому режимі.

Ліхтарик, керований за допомогою кнопкового вимикача, може бути або включений, або виключений за умови, що він перебуває в задовільному

робочому стані. При натисканні кнопки світло запалюється, якщо ліхтарик перебував до цього у вимкненому стані, і навпаки, якщо поточний стан «включений», те після натискання кнопки ліхтарик вимикається. Два стани ліхтарика відповідають двом режимам інтерфейсу. В одному режимі натискання кнопки увімкне світло. В іншому режимі натискання кнопки вимкне світло. Якщо не знати поточний стан ліхтарика, неможливо передбачити, до чого приведе натискання кнопки. Наприклад, якщо він лежить глибоко в сумці. Щоб переконатися, що ліхтарик вимкнений, вам потрібно вийняти його із сумки. Неможливість визначити поточний стан ліхтарика – це приклад класичної проблеми, яка виникає в інтерфейсі, у якому є режими. За станом управляючого механізму неможливо сказати, яку дію варто виконати, щоб досягти поставленої мети. Оперуючи з управляючим механізмом без одночасної перевірки стану системи, користувач не зможе передбачити, який результат буде мати дана операція. Наприклад, натиснуті або вимкнені клавіші Caps Lock і Num Lock приводять до різних режимів. І без погляду на індикатор визначити режим неможливо. На ноутбуках і на деяких клавіатурах таких індикаторів взагалі немає. Кожна людина, яка хоч раз набирала текст може згадати випадок, коли вона замість малих букв набирала заголовні, й потім доводилося перенабрати.

Дональд Норман (1983) указує три методи запобігання *модальних* (тобто пов'язаних з режимами) помилок:

1) не використовувати режими;

2) забезпечувати чітку відмінність між режимами;

3) не використовувати однакові команди в різних режимах, щоб команда, застосована не в тому режимі, не могла привести до неприємностей.

Тема 5. **Когнетика й ергономіка**

5.1. Поняття когнетики й ергономіки

Посібники з розробки продуктів, що взаємодіють з користувачами фізично, звичайно містять конкретну інформацію, засновану на властивостях і

можливостях людського скелету й органів почуттів. Сукупність відомостей у цій області становить науку *ергономіку*.

Ергономіка не вивчає робоче середовище й інші його види безпосередньо, це предмети інших наук. Для ергономіки важливий вплив середовища на ефективність і якість діяльності людини, її працездатність, фізичне й психічне благополуччя. Ергономіка визначає оптимальні величини середовищних навантажень – як за окремими показниками, так і у сполученні.

Вивчення прикладної сфери ментальних здатностей людини називається когнітивним проектуванням, або *когнетикой*.

Деякі когнітивні обмеження очевидні: наприклад, не можна очікувати від звичайного користувача здатності перемножувати в розумі тридцятирозрядні числа за п’ять секунд, тому нема рації розробляти інтерфейс, який вимагав би від користувача такої здатності. Однак розробники часто не враховують інші ментальні обмеження, які впливають на продуктивність користувача при роботі з інтерфейсами «людина-машина», хоча ці обмеження властиві кожній людині.

5.2. Як людина бачить

У людини є 2 типа зору – *центральне* (відповідає за центральну ділянку зорового простору) *і периферичне* (відповідає за бічні ділянки зорового простору). Наші древні предки вижили завдяки периферичному зору. І сьогодні його значення залишається дуже важливим. Дивлячись на екран комп'ютера, люди задіють периферичний зір і звичайно приймають рішення, на якій сторінці зупиниться, на підставі першого враження, яке дає їм периферичний зір. При швидкому погляді людина краще розпізнає об'єкти й більше звертає увагу на те, що вона встигла помітити периферичним зором. Тому, якщо потрібно, щоб користувачі були зосереджені на центральній частині екрана, не варто використовувати анімацію або миготливі елементи на периферії.

Розпізнавання об'єктів. Люди знаходять основні знайомі форми у всьому, що бачать, і використовують ці основні форми, які звуться геометричними іконками (*геонами*), для розпізнавання об'єктів. Ірвін Бідерман у 1985 році

припустив що існує 24 розпізнавані базові форми, з них формуються блоки для побудови всіх об'єктів, які люди бачать й ідентифікують.

Як видно на рис. 5.1 досить просто одержати образи, які впізнаються, групуючи геони. Люди добре реагують на такі зображення. Особливо актуальне використання подібних зображень для створення іконок, де сильно обмежений простір.

Розпізнавання осіб не вписується в теорію геонів, тому що за обличчя відповідає інша частина мозку. Відвідувачі веб-сторінок розпізнають особи й реагують на них швидше, ніж на що-небудь інше. Особливо сильне враження виявляють очі. Якщо очі на веб-сторінці дивляться на оголошення або на продукт, відвідувач також прагне глянути на цей продукт. Однак це не означає що він приділить цьому продуктові особливу увагу.

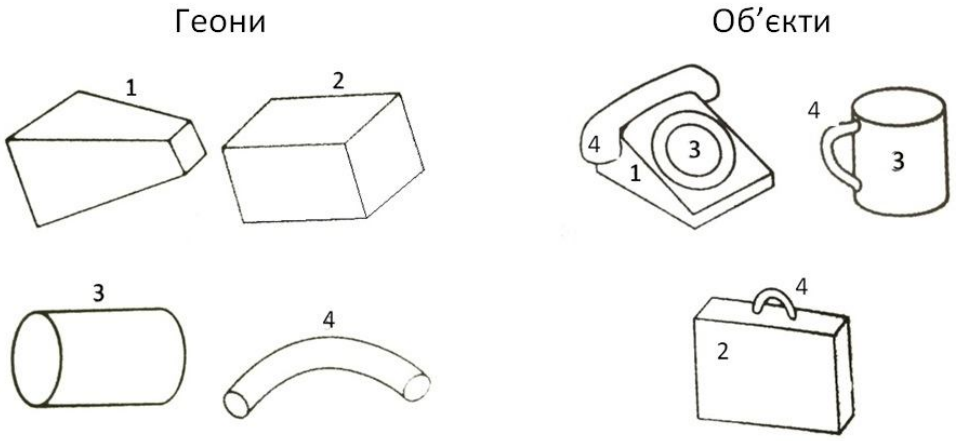


Рис. 5.1 Приклади геонів і формування на їхній основі об'єктів

Двовимірні елементи переважніше тривимірних, тому що очі передають мозку інформацію у двовимірній проекції.

Люди розпізнають малюнки або об'єкти швидше й запам'ятовують їх краще, якщо вони представлені в канонічній перспективі у фронтальній проекції (рис. 5.2а). Тому значки, логотипи й піктограми потрібно малювати саме в такій проекції.

а) б)

Рис. 5.2 Зображення чашки у фронтальній (а) і горизонтальної (б) проекціях

Хоча щодня людина дивиться на чашку зверху, коли п'є з неї (рис. 5.2б), впізнаваною чашка є у вигляді як показано на рис. 5.2а.

Не варто сподіватися, що люди побачать дещо на розробленій веб-сторінці тільки тому що воно там є. Людина зверне увагу тільки на об’єкт, який залучить її увагу в перший момент. Якщо розробник хоче бути впевненим, що люди помітять видимі зміни, потрібно додати візуальні (миготіння) або звукові (гудок) сигнали.

Згідно статистиці 9% чоловіків і 0,5% жінок дальтоніки. Що цікаво, камуфляж для них не є маскуванням. Труднощі викликають розпізнавання червоного, жовтого й зеленого кольорів. Оцінити як виглядає веб-сторінка для дальтоніка можна за допомогою сайтів www.vischeck.com, colorfilter.wickline.org.

Якщо кольори використовуються з певною метою (наприклад зелені кольори обрані для залучення уваги), то бажано для дальтоніків застосувати додаткову схему кодування (наприклад укласти елемент у прямокутник).

А краще підбирати кольори, які розрізняються усіма, наприклад відтінки коричневого й жовтого.

5.3. Як людина читає

Є теорія, за якою малі букви легше читати чим заголовні. Це пояснюється тим, що при читанні ми впізнаємо форми слів і груп слів. Всі слова написані заголовними буквами мають форму прямокутника, а форми слів, написаних прописними буквами індивідуальні (рис 5.3).

Рис. 5.3 Форма слова, написаного прописними й заголовними буквами

Але насправді такі дослідження ніколи не проводилися. Проте Кеннет Паап і Кейт Райнер у 1984 показали, що насправді під час читання відбувається розпізнавання букв, і тільки на цій підставі людина розпізнає слово.

Заголовні букви дійсно читаються повільніше, тому що в людей більше практики в читанні саме текстів складених із малих букв.

До того ж люди звикли, що текст, який складається тільки із заголовних букв вимагає додаткової уваги й сприймається як «крикливий». Тому заголовні букви варто застосовувати, наприклад, при видаленні важливого файлу.

При читанні здається, що погляд переміщується беззупинно, але насправді очі рухаються невеликими стрибками від 7 до 9 букв, які називаються саккадами. Паузи між ними називаються фіксаціями (тривалість 250 мілісекунд). Більшу частину часу погляд не міняє напрямок, але на короткий проміжок часу вертається назад. Крім цього йде ще випереджаюче читання, на 15 символів уперед, тільки для розпізнавання букв.

Людина віддає перевагу коротким рядкам, але швидше читає довгі, тому що кожного разу наприкінці рядка перериваються саккади й фіксації. Переваги людей використовуються при верстці газет і журналів.

Варто використовувати довгі рядки (100 символів у рядку), якщо важлива швидкість читання, короткі рядки (45-72 символів), якщо швидкість читання менш критична. Для багатосторінкових статей краще використати безліч колонок з невеликим рядком (45 символів).

Під час читання людина не займається точним ототожненням букв і слів; вона інтерпретує їх пізніше. Людині властиво вгадувати, що буде далі. Чим більше людина знає, тим легше вгадувати й інтерпретувати.

Хчоа свола зімнени, все-таки ви мжеоте пчиорати яикомсь чниом цей азабц. Але пршеа й отнасня бувки кдножго солва пвиноні бути на сївох міцях. Ршета букв мжуоть бути пешареміні й все-таки ви проєтачите тескт без виеклих зсиуль. Це тмоу що чинntая овасноно на вагувадnні нансптуого свлоа.

Для розуміння прочитаного дуже важливим є заголовок. Наприклад, маємо текст: «Для початку розділите елементи на групи. Поділ на основі кольорів є загальноприйнятим, але можна також використати й інші критерії, такі як текстура, матеріал або режим обробки, зазначений у супровідній документації. Після сортування приступайте до використання встаткування. Кожна група обробляється окремо. Після сортування помістіть одну з них у машину.»

Про що цей абзац зрозуміти важко, поки не додали *заголовок*: «Як використовувати пральну машину». Зручність читання не покращилася, проте хоча б стало зрозуміло про що цей абзац.

З огляду на особливості людського бачення й читання можна зробити наступні виводи:

- те, що розуміє й запам'ятовує читач залежить від його попереднього досвіду, відношення до прочитаного й інструкцій, які були дані йому перед цим;
- імовірність, що користувачі запам'ятають конкретну інформацію невелика;
- текст варто спорядити значимими заголовками;
- завжди пам'ятати про цільову аудиторію. Якщо текст призначений для широкого кола читачів, то варто використовувати прості слова.

5.4. Як людина думає

Наприклад, людина оплачує рахунки через веб-сайт. У процесі розв’язку цього завдання вона думає й згадує (когнітивні процеси), дивиться на екран (візуальний процес), натискає кнопки й пересуваєте мишу (моторні процеси). Все це пов'язане з відповідним видом навантажень. Порядок розташування навантажень від більше витратної до менш витратного наступний:

- когнітивна;
- візуальна;
- моторна.

Тому при розробці інтерфейсів варто створювати компромісні рішення, перемежаючи види навантажень. Кілька додаткових щигликів кнопкою будуть прекрасною альтернативою розумовим зусиллям користувача. У той же час, якщо зосередитись тільки на моторному навантаженні це теж не доведе до добра. Варто мінімізувати моторні перемикання. Наприклад, якщо людина володіє методикою сліпого набору, то перемикання між клавіатурою й мишею буде її дратувати. Необхідно передбачати гарячі клавіші.

Хоча моторне навантаження менш «затратне», його теж бажано знизити. Потрібно переконатися, що об'єкт, на якому користувачеві потрібно клацнути, не занадто малий і перебуває не занадто далеко. Не варто змушувати користувача переміщувати курсор через весь екран, щоб клацнути на кнопці або на маленькій стрілці у меню, що випадає, з якого вибирається потрібний пункт.

Формула, яка описує наскільки великою повинна бути ціль, щоб користувач рухаючи курсор по екрану легко попадав у неї, називається *формулою Фітса*.

$$T = a + b\log_2\left(1 + \frac{D}{W}\right)$$

Тут T – середній час, затрачуваний на здійснення дії;

a – середній час запуску/припинення руху (час затримки);

b – величина, що залежить від типової швидкості руху;

D – відстань від стартової точки до центра цілі;

W – ширина цілі, виміряна уздовж осі руху.

Перед тим, як перемістити курсор до цілі, або зробити будь-яку іншу дію з набору безлічі варіантів користувач повинен вибрати цей об'єкт або дію. У *законі Хіка* затверджується, що коли необхідно зробити вибір з n варіантів, час на вибір одного з них буде пропорційним логарифму за основою 2 від числа

варіантів плюс одиниця, за умови, що всі варіанти є рівноімовірними. У цьому виді закон Хіка дуже схожий на закон Фітса:

Якщо ймовірність першого варіанту дорівнює *p(i)*, то закон Хіка має вигляд:

Коефіцієнти, які використовуються у виразі закону Хіка, у великій ступені залежать від багатьох умов, включаючи те, як представлені можливі варіанти, й те, наскільки добре користувач знаком із системою. Якщо варіанти представлені незрозумілим чином, значення *a* і *b* зростають. Наявність навичок і звичок у використанні системи знижує значення *b*.

Користувачі дуже не люблять чекати, особливо коли вони не знають, скільки часу може зайняти очікування. Якщо людина знає, що процес займає три хвилини, вона можете піти випити чашечку кави й повернутися. Якщо процес займає одного разу 30 секунд, іншого 5 хвилин, то ці 5 хвилин здадуться вічністю. Тому завжди потрібно використовувати індикатори ходу процесу, це допоможе зрозуміти, скільки залишилося часу до завершення. Наскільки це можливе треба робити так, щоб кількість часу, який затрачується на виконання завдання або одержання інформації, відповідала очікуванням людей. З метою імітування, що процес відбувається швидше, його варто розбити на кроки, і менше змушувати людей думати. Чим більше інформації треба обробляти, тим більше довгим здається процес.

*Щоразу, пропонуючи опцію, розробник просить користувача зробити вибір***.** Тобто користувачам потрібно буде щось обмірковувати й щось вирішувати. Це не обов'язково погано, але загалом, варто завжди мінімізувати кількість питань, по яких користувач повинен прийняти рішення. Сама по собі ідея, просити користувача зробити вибір, не така вже і погана. Право вибору гарантоване конституцією й іноді навіть надає задоволення (наприклад яке

морозиво вибрати – шоколадне чи полуничне). Проблеми виникають, коли людей просять зробити вибір, який їх *нітрохи не хвилює* (рис. 5.4).

Наприклад користувач хоче одержати допомогу, викликає довідку й бачить таке вікно

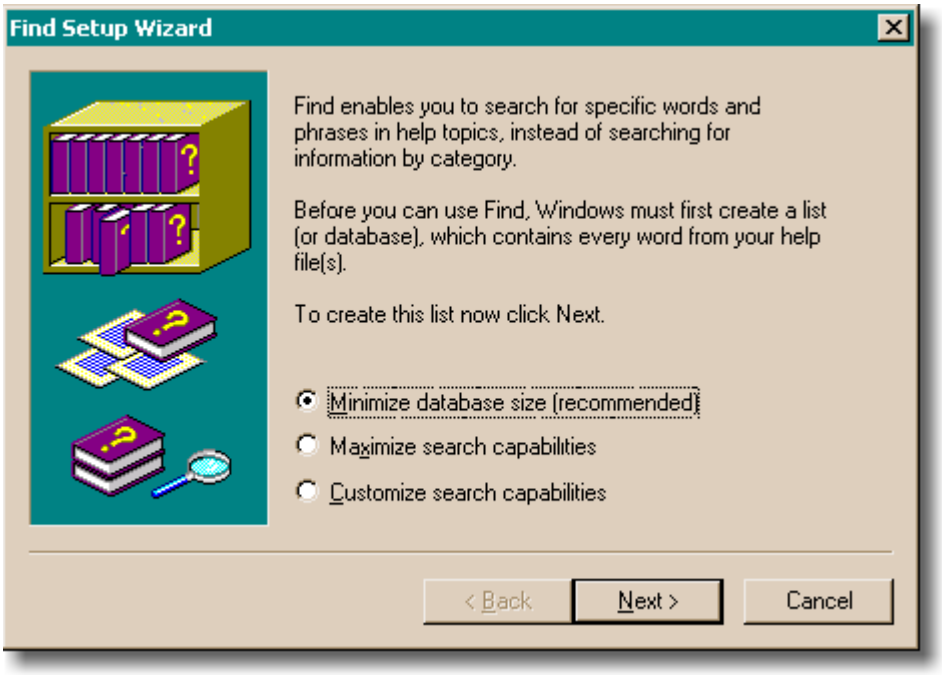


Рис 5.4 Вибір, який не хвилює користувача

Це вікно, видане за результатами пошуку. Тут сказано, що Windows зараз зробить список знайденого за ключовими словами, й користувачу пропонується вибрати розмір цього списку (бази даних): мінімальний, максимальний чи визначити розмір самостійно. Проблема перша: це вікно відволікає. Користувач намагається знайти *допомогу* у файлі *допомоги*. У цей конкретний момент йому абсолютно однаково, чи база даних маленька, велика, оптимізована, чи вона під потреби клієнта або покрита шоколадом. Йому потрібна *допомога*, а цей огидний діалог повідомляє, що система повинна створити список або базу даних. І педантично читає лекцію на цілих три параграфи, зміст яких приводить користувача у здивування.

Більшість людей, що звертаються до допомоги, просто не розуміють цих ієрогліфів. Швидше за все навіть просунуті користувачі, програмісти з кандидатським ступенем, які знають усе про побудову оптимізованих бінарних деревоподібних структур при індексуванні текстів, не зможуть зрозуміти, а

який, властиво, вибір їм пропонують зробити і не нашкодить цей вибір результатам пошуку?

Це не означає, що потрібно *повністю* позбавити користувача вибору. Йому часто доведеться ухвалювати рішення щодо того, як буде виглядати його документ, як повинна реагувати його веб-сторінка, і про роботу, над котрою він у конкретний момент працює. Немає нічого краще, за те, ніж надати користувачеві в цих областях право приймати рішення – і чим більше, тим краще. Є ще одна область, де можливість вибирати робить задоволення: зміни зовнішнього вигляду, які не приводять до змін у функціональності. Кожний з нас міняє заставки робочого стола. Оскільки подібні опції міняють зовнішній вигляд без зміни функціональності й користувачі можуть без ризику для себе їх проігнорувати й спокійно працювати, використання подібних опцій тільки вітається.

Якщо розробник створює мультікультурний продукт, краще провести дослідження аудиторії в декількох географічний регіонах. Якщо типовим представникам заходу показати картинку, вони зосередять свою увагу на головному або домінуючому об'єкті на передньому плані, у той час як жителі сходу звернуть увагу на контекст і фон. Те ж саме стосується сприйняття колірної гами, умовних позначок і т. п.

Третину часу наші думки десь бродять. Люди можуть зосередитися на одному завданні протягом обмеженого часу. Тому варто переконатися, що у додатку не забуто про зворотний зв'язок, який допомагає користувачам зрозуміти, де вони перебувають, оскільки якщо їхній розум десь блукав, вони повинні швидко зорієнтуватися. Особливо це стосується веб-сайтів.

Людям подобається розподіляти речі за категоріями. Якщо є деяка кількість інформації й вона не організована за категоріями, то люди почувають себе загубленими й намагаються організувати інформацію своїм власним способом. Важливо не те, хто організував інформацію, а те наскільки добре вона організована. Розробнику треба спробувати зрозуміти яка схема організації близька й зрозуміла користувачам. Виключення становлять діти.

Якщо створюється програмне забезпечення для дітей молодше 7 років, будь-яка організація за категоріями буде мати сенс для дорослих, які перебувають поруч із дітьми, а не для них самих.

Для людини більше зрозумілі не абсолютні значення, а відносні, добре якщо вони представлені ще й візуально.

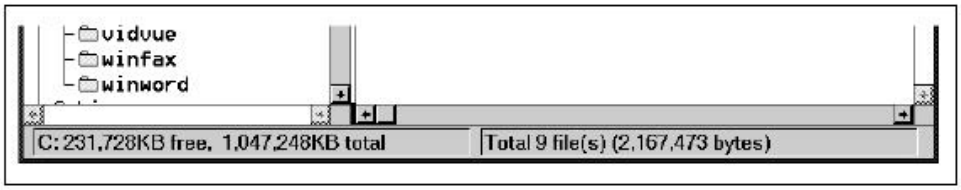


Рис. 5.5 Приклад використання абсолютних значень в інтерфейсі

Так наприклад абсолютне значення вільного місця на диску змушує задуматися (рис. 5.5).

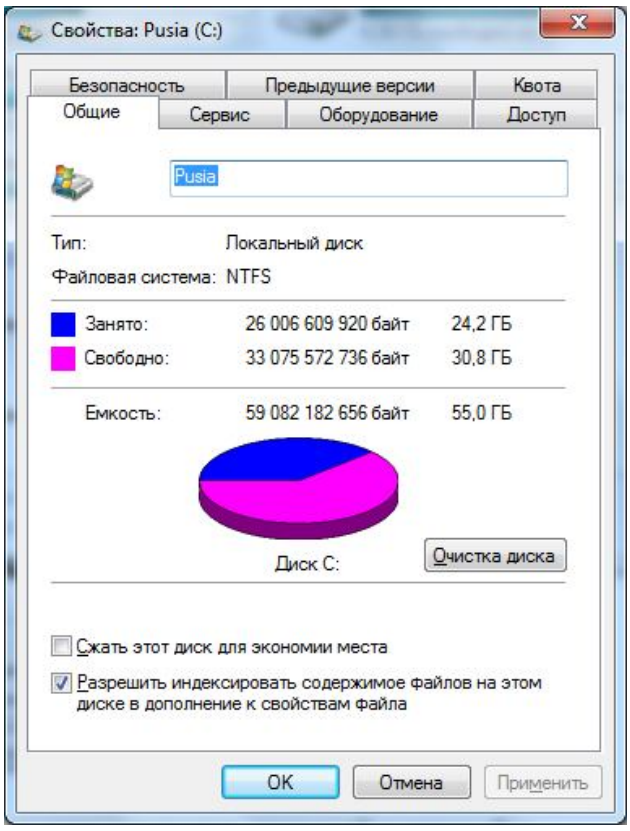


Рис. 5.6 Приклад використання відносних значень в інтерфейсі

А якщо сказати що вільний простір 20% диску, те це відразу створює деяку картинку у мозку(рис. 5.6).

5.5. Когнітивний опір

Згідно одному з найперших принципів дизайну графічних інтерфейсів користувач не повинен запам'ятовувати те, що може запам'ятати комп'ютер.

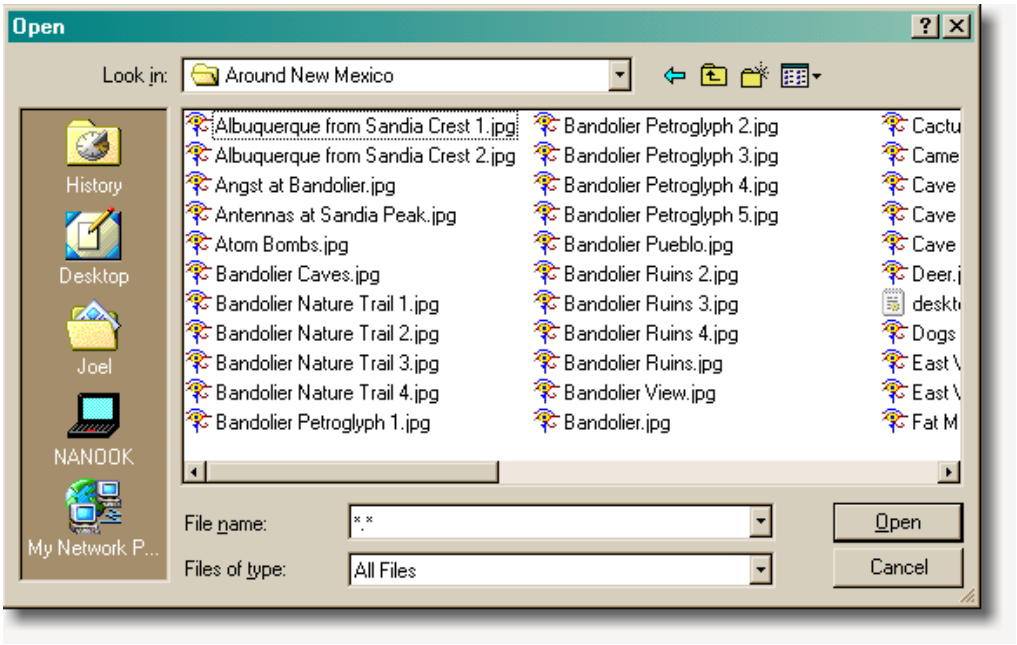


Рис. 5.7 Діалог відкриття файлу в *Windows 95*

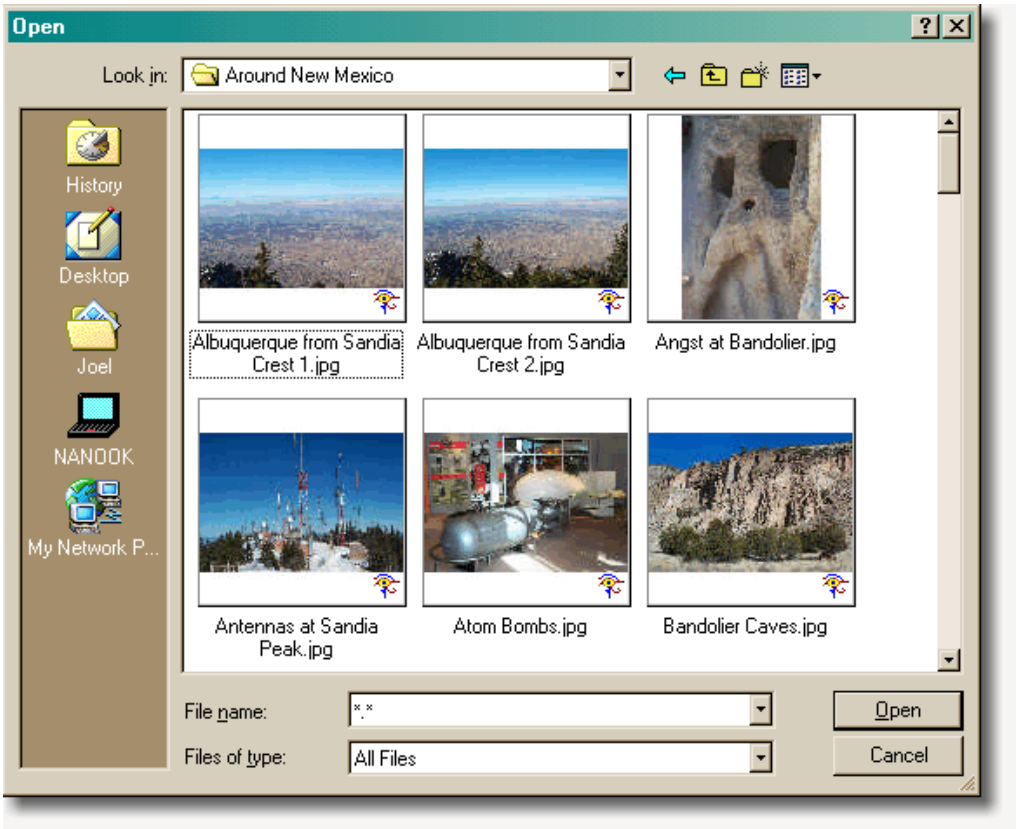


Рис. 5.8 Діалог відкриття файлу в *Windows 98*

Класичний приклад – діалогове вікно *«Відкрити файл»*, яке пропонує користувачеві вибрати один файл зі списку, а не змушує вводити точне ім'я

файлу по пам'яті. Людина набагато легше пригадує що-небудь, коли в неї є підказки, і завжди віддасть перевагу вибору зі списку замість того, щоб вивуджувати необхідну інформацію із глибин своєї пам'яті.

Іншим прикладом є самі списки файлів. До *Windows 98* картина була така, як на рис. 5.7. На щастя, в *Windows 98* увели підтримку попереднього показу картинки, і зараз видно ті ж самі файли у спосіб, показаний на рис. 5.8.

Тепер знайти потрібний файл стало значно легше; від користувача навіть не потрібно ніяких розумових зусиль для того, щоб зіставити слова із зображенням.

Принцип мінімального запам'ятовування використовується також у процесі *інтелектуального завершення уведення* (auto completion). Наприклад, при уведенні тексту деякі програми можуть робити свої припущення про те, що користувач збирається друкувати (рис. 5.9).

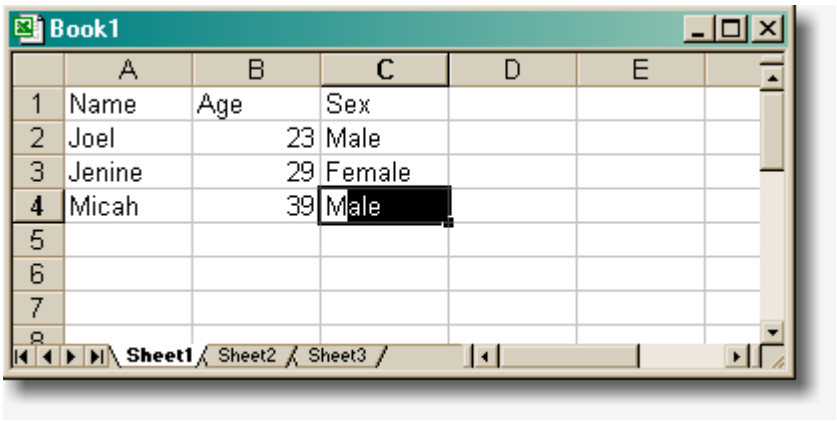


Рис. 5.9 Інтелектуальні підказки в інтерфейсі

Як тільки користувач надрукує «M», Excel запропонує варіант «Male», тому що користувач уже вводив це слово в цій колонці, і Excel може тим самим автоматично завершити процес уведення. Якщо ж користувач хоче набрати «Mystery», а не «Male», то просто ігнорує підказку Excel й уводить потрібне слово, не витрачаючи додаткових зусиль.

Правда, трапляються й проколи, як це міг помітити кожен, хто користувався Microsoft Word англійською у травні місяці (рис. 5.10). Excel не робить різниці між дієсловом могти (*may* – у реченні) й назвою місяця травень (*may* – у підказці).

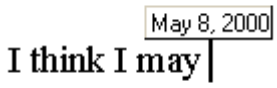


Рис. 5.10 Невдале застосування інтелектуальної підказки

Використання таких термінів, як *свідоме* й *несвідоме*, які мають цілком певне значення в психології, філософії й історії й застосовуються для опису аспектів функціонування нашого мислення, може викликати деякі ускладнення. У контексті технічного проектування має сенс користуватися більше обмеженими поняттями *когнітивного свідомого й когнітивного несвідомого.* Ще більш точними були б терміни *емпіричне свідоме* й *емпіричне несвідоме*, Дамо коротке визначення: *несвідомими* називаються ті ментальні процеси, які людина не усвідомлює в той момент, коли вони відбуваються.

Оскільки міркування про те, що є свідомим або несвідомим, здаються досить далекими від повсякденних турбот людини, спробуємо наочно продемонструвати їхнє значення у звичайному житті. Спробуйте відповістити на наступне запитання: яка остання буква у вашому імені? Доти, поки ви не прочитали попередню пропозицію, ви, імовірно, не думали про цю букву й її зв'язок з вашим ім'ям.

Дана інформація не запитувалася, однак кожна людина зможе її одержати, коли в цьому виникає необхідність. Те місце, звідки була витягнута буква, будемо називається когнітивним несвідомим. Коли ви про це задумалися, буква перейшла у свідомий стан.

Раптовий звук або інша несподівана подія може відволікти увагу людини від того, чим людина була зайнята в цей момент – наприклад, від читання книги, до питання про причину цього звуку. Скажемо, почувши звук падаючої з полки лампи, можна подумати: напевно, це кіт туди заліз. Після того як людина повернеться до читання, знання про подію, яка відбулася, переміститься з когнітивного свідомого у когнітивне несвідоме.

Поки людина вивчає деяке завдання, вона може сприймати його як розгалужену подію, яка вимагає свідомої уваги. У міру повторення завдання його виконання може стати нерозгалуженим й автоматичним.

Таблиця 2. Властивості когнітивного свідомого й когнітивного несвідомого

Характеристика	**Свідоме**	**Несвідоме**
Ініціація	Чимось новим	Повторенням
	Нестандартними ситуаціями	Очікуваними подіями
	Небезпекою	Безпекою
Використання	У нових обставинах	У звичних ситуаціях
Вирішення завдань	Прийняття рішень	Робота з нерозгалуженими завданнями
Прийняття тверджень	Логічних	Логічних або суперечливих
Функціювання	Послідовне	Одночасне
Управління	Волею	Звичними діями
Продуктивність	Невелика	Величезна
Період функціонування	Десятки секунд	Десятиліття (все життя)

Запит комп'ютерної системи, призначений служити як міра безпеки, через звичку стає марним і тільки ускладнює звичайний процес видалення файлів. Вся справа в тому, що *будь-який запит про підтвердження, який вимагає встановленої відповіді, незабаром стає марним*.

Розробники, які використовують такого роду підтвердження, і адміністратори, які думають, що запити про підтвердження забезпечують безпеку, насправді не враховують силу властивості формування звичок, притаманну когнітивному несвідомому. Більше ефективний підхід полягає в тім, щоб дати користувачеві можливість скасувати помилкову команду, навіть якщо після неї були зроблені якісь інші дії.

5.6. Локус уваги

Із всіх об'єктів або явищ навколишнього світу, які сприймаються за допомогою власних почуттів або уяви, у кожний момент часу людина може

сконцентруватися тільки на одному. Чим би не був цей об'єкт, деталь, спогад, думка або поняття, він стає *локусом* уваги людини. У цьому випадку мається на увазі не тільки та увага, яку можна активно звертати на що-небудь, але також і пасивне сприйняття потоку вхідної інформації або просте переживання того, що відбувається.

Розходження між фокусом і локусом уваги можна зрозуміти на прикладі наступного речення: «Ми можемо цілеспрямовано сфокусувати нашу увагу на будь-якому локусі». Тоді як *фокусувати* означає вольову дію, повністю управляти змістом локусу своєї уваги людина не здатна. Якщо людина чує, як за нею раптово вибухнула петарда, її увага буде спрямована на джерело звуку.

Увага вибіркова. Наскільки важко привернути увагу людей залежить від ступеню їхньої зацікавленості. Наприклад, якщо вони зайшли на сайт купити подарунок і не впевнені у своєму виборі, то можна залучити їхню увагу анімацією, картинками. З іншого боку, якщо людина зосереджена на заповненні полів, те вона ігнорує всі сигнали, що не відносяться до цього завдання, у тому числі відео й картинки.

Підсвідома увага людей постійно сканує навколишній світ з метою виявлення певних речей. Є речі, які споконвічно хвилюють людину – власне ім'я, послання пов’язані з їжею, водою, небезпекою.

Людям притаманно фільтрувати інформацію. Напевно всі зустрічали «впертих» людей, які плекають свої переконання й не хочуть їх міняти незважаючи на очевидні доводи. Для такої людини має цінність тільки інформація, яка підтверджує її правоту. А от протилежна інформація буде ігнорована, знецінена й розкритикована. Це властиво для всіх людей.

Надія на те, що люди звернуть увагу на представлену розробником інформацію може виявитися даремною. При проектуванні не варто робити припущення. Те, що очевидно для вас, як для розробника, може виявитися неочевидним для користувачів.

З врахуванням того, що люди будуть фільтрувати інформацію, потрібно використовувати кольори, розмір, анімацію, відео й звук, щоб привернути увагу до важливих елементів.

Якщо потрібно, щоб люди звернули увагу на певну інформацію, вона повинна відображатися на екрані протягом часу в 10 разів більше, ніж вам це здається необхідним.

Безперервність уваги зберігається біля 10 хвилин. Якщо інформація цікава, то можна удержати увагу протягом 7-10 хвилин. Якщо потрібно удержати увагу довше, то варто звернутися до нової інформації або зробити перерву. Це правило варто враховувати при розміщенні великих статей, аудіо або відеороликів на сторінках сайту.

Людина звертає увагу тільки на *помітні сигнали*. Але для кожного ці сигнали свої. Так на грошах зазвичай люди дивляться на цифри, якщо зроблять зусилля, то згадають чиї там портрети, а от які міста на кожній купюрі, не говорячи вже про більш дрібні знаки й відмінності люди не пам'ятають. Проте для нумізмата все буде сигналом – і рік випуску, і підпис і т. п.

При розробці інтерфейсу варто вирішити, які сигнали будуть помітними для цільової аудиторії й зробити так, щоб вони були очевидними.

Людина не пристосована до багатозадачності. Проблема телефонних розмов за кермом не в тому, що телефон у руці, а в самій розмові, таким чином використання блютус-гарнітури не вирішує проблеми аварійності на дорогах з вини телефону.

З тих самих причин не варто змушувати користувачів виконувати кілька завдань одночасно: розмовляти з клієнтом і заповнювати форму на комп'ютері. Якщо люди все ж таки повинні це робити одночасно, потрібно приділити особливу увагу тому, щоб форма була зручної або в ній легко можна було виправити помилки.

Мовою когнітивної психології будь-яке завдання, яке людина навчилася виконувати без участі свідомості, стає *автоматичним*. Автоматизм дозволяє виконувати відразу кілька дій одночасно (наприклад, іти й розмовляти). Всі

завдання, які виконуються одночасно, за винятком не більш ніж одного, є автоматичними. Те завдання, що не є автоматичним, природно, перебуває безпосередньо в локусі уваги.

Людина може бути в різному ступені поглинена завданням, яке у цей момент перебуває в локусі її уваги. Чим більше вона зосереджена на завданні, тим сутужніше їй перейти на інший локус уваги, й тим більше сильний стимул потрібний для того, щоб такий перехід відбувся.

Інтерфейси варто розробляти з розрахунком на те, що користувач, поглинений своїм завданням, не стане навіть реагувати на спроби програмного продукту поспілкуватися з ним. Інтерфейс повинен добре працювати незалежно від ступеню захопленості користувача. Наприклад, розроблювачі інтерфейсів іноді вважають, що локус уваги користувача пов'язаний з курсором, і що за допомогою зміни форми курсору можна управляти увагою користувача. Хороше місце для розміщення вказівників – це місце розташування курсору, але навіть там вони можуть залишитися непоміченими. Форма курсору не є локусом уваги користувача, – скоріше їм може бути місце або об'єкт, на який курсор указує.

Чим більше критичним є завдання, тим менше ймовірність того, що користувач помітить попередження щодо тих або інших потенційно небезпечних дій.

Попереджуюче повідомлення з найбільшою ймовірністю може залишитися непоміченим саме в той момент, коли інформація, яка втримується в ньому, має найбільшу цінність.

Наявність у людини тільки одного локусу уваги має й позитивні сторони. Фокусники експлуатують цю властивість психіки самим безсоромним чином. Вмілий фокусник може зафіксувати увагу всієї аудиторії на одній руці, і жоден глядач не помітить того, що в цей час робить інша рука фокусника, хоча вона ніяк і не схована.

В інтерфейсах цим користуються, щоб сховати різного роду запізнювання. Наприклад, карткова гра, у якій на генерацію кожної нової роздачі потрібно

кілька секунд, здається більше швидкої, якщо в цей час відтворювати шелесткий звук карт, що тасуються. Цінність такого маскування виявляється, якщо цей звук раптово відключити. Тоді затримка відразу стане здаватися дратівною.

Поняття локусу уваги дозволяє точно визначити поняття побічного ефекту. *Побічний ефект* – це наслідок застосування команди, при якому змінюється зміст або події, які не перебувають у локусі уваги людини. Наприклад, коли людина вставляє раніше скопійований текст на місце виділеного, у локусі уваги перебуває текст, що вставляється, а побічним ефектом є видалення. Усунення побічних ефектів повинне бути однією із цілей для розробника людиноорієнтованого інтерфейсу.

5.6. Короткочасна й довгострокова пам'ять

Сприйняття не завжди відкладаються в пам'яті. Більшість сприйняттів втрачаються після того, як загасають. З погляду розробки інтерфейсів зі швидкого загасання сенсорних сприйняттів надходить, що людина, яка прочитала або почула 5 секунд назад деяке повідомлення, необов'язково зможе згадати його зміст. Якщо таке повідомлення важливо саме по собі або містить важливу деталь, наприклад, номер у повідомленні «Помилка 39-152», то воно повинно залишатися на екрані до тих пір, поки не перестане бути актуальним (такий підхід можна назвати найкращим), або ж необхідно надати користувачеві можливість негайно обробити цю інформацію, перш ніж вона зникне з його пам'яті.

Дитячі спогади, номери телефонів, імена старих друзів – все це перебуває в *довгостроковій* пам'яті людини, звідки цю інформацію можна дістати за бажанням.

Коли деяка інформація стає локусом уваги, вона переміщується у *короткочасну пам'ять*. Там вона буде зберігатися протягом десяти секунд.

Погано вимагати від користувача запам'ятовувати інформацію в одному місці й переносити в інше, наприклад, читати букви або цифри на одній

сторінці й потім уводити в іншій, – вони можуть забути інформацію й будуть у розпачі.

Якщо розробник пропонує людині «занести» що-небудь у робочу пам'ять, головне не відволікати її, поки вона повністю не розв’яже це завдання. Робоча пам'ять чутлива до перешкод – велика кількість сенсорних сигналів перешкоджає фокусуванню уваги.

Користувачі, особливо новачки, звичайно віддають перевагу багатої кількості дрібних операцій перед однією великою операцію, тому що в цьому випадку вони можуть не тільки краще контролювати загальне просування розв’язку й забезпечити його задовільний хід, але й відволіктися від деталей роботи на попередніх етапах.

Наявні результати деяких досліджень дозволили розробити наступні рекомендації із припустимого часу відповіді інтерактивної системи:

- 0,1...0,2 с – для підтвердження фізичних дій (натискання клавіші, робота зі світловим пером, мишею);
- 0,5...1,0 с – для відповіді на прості команди (наприклад, від моменту уведення команди, вибору альтернативи з меню до появи нового зображення на екрані);
- 1...2 с – при веденні зв'язного діалогу (коли користувач сприймає серію взаємозалежних питань як одну порцію інформації для формування однієї або декількох відповідей, затримка між наступними питаннями один за одним не повинна перевищувати зазначену тривалість);
- 2...4 с – для відповіді на складний запит, який полягає в заповненні деякої форми. Якщо затримка не впливає на іншу роботу користувача, пов'язану з першою роботою, затримки до 10 с можуть бути прийнятними;
- більше 10 с – при роботі в мультизадачному режимі, коли користувач сприймає дане завдання як фоновий процес.

Прийнято вважати, якщо користувач не одержує відповідь протягом 20 с, то це не інтерактивна система. У такому випадку користувач може «забути»

про завдання, зайнятися розв’язком іншого завдання й повернутися до нього тоді, коли користувачу буде зручно. При цьому програма повинна повідомляти користувачеві, що затримка відповіді не є наслідком виходу системи з ладу (наприклад, шляхом регулярного відновлення рядка стану системи або ведення протоколу виконання завдання користувача).

5.8. Як працює пам'ять

Людина одночасно може запам'ятати тільки чотири елементи. Одна зі стратегій, яка застосовується для запам'ятовування більшої кількості інформації – це поділ на групи, наприклад для запам'ятовування телефонних номерів. Добування інформації з пам'яті теж підкоряється правилу чотирьох.

Таблиця 3. Слабкі й сильні сторони людей і комп'ютерів

Сильні сторони	Слабкі сторони
Люди	
• розпізнавання образів; • перемикання уваги; • нескінченна ємність довгострокової пам'яті; • здатність до навчання.	• короткострокова пам'ять із малою ємністю; • швидка втрата даних з короткострокової пам'яті; • повільна обробка даних; • помилки; • ускладнений доступ до довгострокової пам'яті.
Комп’ютери	
• пам'ять із великою ємністю; • довгострокова пам'ять; • висока швидкість обробки; • обробка без помилок; • безвідмовний доступ до пам'яті.	• просте порівняння з еталоном; • обмеження здатності до навчання; • обмежена ємність довгострокової пам'яті; • обмежена інтеграція даних.

Тому при організації інформації на екрані її варто розділяти за категоріями і намагатися дотримувати правило 4-х. Але, якщо завдання в ці рамки ніяк не вміщується – не смертельно, адже люди не завжди покладаються тільки на власну пам'ять, вони використовують записні книжки.

Вивчати інформацію легше, якщо вона укладається вже в існуючу *схему*. Наприклад, голова людини складається з десятків елементів, але ми їх всі пам'ятаємо, тому що вони укладаються в певну схему. Якщо в людей уже є схема, яка має відношення до інформації, котру розробник їм хоче повідомити, йому треба переконатися, що він має достатнє уявлення про цю схему.

Інформацію легше розпізнати, чим згадати. При розробці важливо намагатися зменшувати навантаження на пам'ять. Правильний дизайн дозволяє уникнути проблем, пов'язаних із улаштуванням людської пам'яті.

Людина схильна забувати. Якщо інформація не перебуває в довгостроковій пам'яті, то нова інформація забудеться на 50% через день, на 80 % через тиждень, на 90% через місяць, а через рік буде важко згадати про саму подію.

Дизайн повинен враховувати цю здатність. Якщо інформація є важливою, треба надати можливість легко звертатися до неї знову й знову.

Звичайно, перервавши деяку роботу, користувач потім до неї вертається. Якщо перерва триває всього декілька секунд, у межах періоду загасання короткочасної пам'яті, додаткових стимулів, для того щоб людина могла повернутися до виконання поточного завдання, не потрібно. Якщо період більш тривалий, то повернення до виконання перерваного завдання повинне бути чимось ініційовано – наприклад, видом незакінченої роботи, яка лежить перед людиною.

Якщо користувачі не можуть зрозуміти інформацію на екрані й запросять довідку за цією темою – не можна дозволити вікну довідкової системи закрити ту інформацію, для якої вона викликана. Подібна допомога називається *деструктивною*, тому що закриває той предмет, на якому користувачеві потрібно акцентувати увагу. Користувачі звичайно прибігають до довідкової системи два-три рази, поки повністю не сприймуть інформацію.

Проектування користувацького інтерфейсу базується на знанні того, як людина пізнає й сприймає. Одне з найбільш важливих завдань інтерфейсу: зменшувати довіру користувача до власної пам'яті й використовувати переваги комп'ютеру для підтримки людських слабостей.

5.9. Ментальні моделі.

Ментальна модель являє собою розумовий процес, спрямований на розуміння того, як працює що-небудь (тобто розуміння навколишнього світу). Ментальні моделі засновані на неповних фактах, минулому досвіді й навіть на інтуїтивному сприйнятті. Вони допомагають формувати дії й поводження людини, впливають на її поведінку у складних ситуаціях (на що звертається увага), і визначають підхід людини до проблем.

Ментальні моделі дозволяють:

- передбачати (або відзначати невидимі) події;
- знаходити причини помічених подій;
- визначати необхідні дії для здійснення потрібних змін;
- використовувати їх як мнемонічні пристрої для запам'ятовування подій і зв'язків (відносин);
- забезпечувати розуміння аналогічних пристроїв;
- застосовувати стратегії, які дозволяють перебороти обмеження, закладені в алгоритмі обробки інформації.

Ментальні моделі створюються дуже швидко, раніше, ніж відбувається реальне спілкування користувача з програмним забезпеченням або пристроєм. Модель відображає очікування людини, яка працює з комп'ютером, і той досвід, який вона одержує в результаті.

Ментальні моделі змінюються. Люди звертаються до них, щоб передбачувати, як система, програма або продукт будуть поводитися, і для того, щоб зрозуміти, що з ними робити.

Розуміння ментальних моделей цільової аудиторії – запорука успіху сайту або іншого продукту.

При розробці користувацького інтерфейсу відбувається зіткнення трьох моделей – власне користувача, програміста й проектувальника.

На одне й те ж саме програмне забезпечення користувач дивиться з погляду свого досвіду з реального миру й інших програм, у нього є уява, як повинні виглядати завдання, процеси, інструменти й результати. Програміст опирається на інструменти розробки, принципи й методи програмування, операційну систему. Проектувальник інтерфейсу повинен розуміти концептуальну модель користувача, модель програміста, а так само методи проектування користувацького інтерфейсу й у такий спосіб виступити сполучною ланкою між програмістом і користувачем.

Модель користувача. Коли користувач приступає до роботи з новою програмою, його голова наповнена досвідом минулих зустрічей з комп'ютером. У нього є певні очікування із приводу того, як нова програма буде працювати. Якщо він уже використовував подібне програмне забезпечення, він буде думати, що ця нова програма буде працювати подібним чином і відповідати якимось загальним визначеним умовностям. У нього навіть можуть бути цілком розумні думки про те, як буде працювати інтерфейс даної програми. Все це називається моделлю користувача – уява про те для чого і як програма буде працювати.

Більшість дебатів із приводу користувацьких інтерфейсів зовсім зайві. Суть не в тому, що одна операційна система краще іншої, тому що пропонує більше способів міняти розмір вікна. Суть у наступному: чи реагує користувальницький інтерфейс так, як користувач того очікує? Якщо ні, користувач буде відчувати власну безпорадність і неможливість контролювати ситуацію.

Хороший дизайн користувальницького інтерфейсу має на увазі, що програма відповідає очікуванням користувачів про те, як вона повинна поводитися. Поганий дизайн інтерфейсу звичайно викликає в користувачів сумнів у правильності своїх дій і спричиняє дивне, а іноді й марновірне поводження, якщо звичайно це не програма для дітей, які дуже охоче досліджують інтерфейс. Інтерфейс для малят проектується так, щоб об'єкти й

предмети, які рухаються, зацікавлювали маленьких користувачів. Діти допитливі й клацають мишкою по всьому, що бачать на екрані.

Модель програміста – найлегша для відображення, тому що вона може бути формально й недвозначно описана. Насправді дана модель є представленою в певному виді функціональною специфікацією програмного продукту. Програміст – людина, яка кодує програму або систему.

Об’єкти й дані, складові програми цікаві програмістові, але не обов'язково в плані того, як користувач взаємодіє з інформацією. Наприклад, з погляду програміста, інтерфейси, призначені для збереження й відновлення інформації, являють собою поля даних або записи в базі даних. Точка зору на них у користувача може бути іншою, ніж у розробників. Ті самі дані можуть бути входом у програму для перевірки, особисту записну книжку або ділову телефонну книгу.

Програміст знає комп'ютерну платформу, операційну систему, інструменти розробки, керівництва й специфікації з програмування потрібні для створення програм. Таким чином, тут не враховується вміння програміста надавати користувачеві найбільш прийнятні моделі й метафори користувальницького інтерфейсу. Як сказав програміст Эд Кеннеді: «Це була б велика програма, якби не було всіх цих проклятих користувачів».

Модель проектувальника описує об'єкти, з якими працює користувач і техніку маніпулювання ними. Складові моделі проектувальника:

- *подача інформації* – містить у собі кольори, анімацію, звук, форму об'єктів, графіку, текст, розташування інформації на екрані. Це часто найбільш видима, але не найважливіша складова. Кольори можуть спочатку зацікавити користувача, але потім відволікати від роботи. Ця частина роботи проектувальника становить 10%;
- *взаємодія, відчуття інтерфейсу* – включає дії із клавіатурою, функціональними клавішами й іншими пристроями введення, зворотний зв'язок з користувачем. Це теж видима частина. Але у відмінності від першої вона саме

відповідає за зручність використання. Тому становить 30% моделі проектувальника;

- *взаємозв'язок об'єктів* – включає властивості об'єктів й їхній зв'язок. Відповідає за підбір метафор, для того щоб погодити завдання, які повинна виконувати людина з можливостями їхньої реалізації комп'ютером. Це тривала робота, яка і становить останні 60% моделі проектувальника. Успіх взаємодії й подачі інформації залежать саме від цієї підготовчої роботи, яка у готовому програмному продукті непомітна.

У результаті виходить модель програми, яка також називається *концептуальною моделлю*. Тобто діюча модель, яку людина одержує при знайомстві з дизайном й інтерфейсом конкретного продукту. Концептуальну модель продукту надає саме інтерфейс.

Розглянемо приклад. При додаванні картинки в документ Microsoft Word (або будь-якого іншого текстового редактора), картинка зберігається в тому же файлі, що й сам документ. Можна створити картинку, вставити її в документ, *видалити оригінальний файл із картинкою й однаково* картинка збережеться в документі.

HTML такого не дозволяє. Картинки в HTML повинні зберігатися в окремих файлах. Якщо посадити перед симпатичним WYSIWYG (what I see is what I get – що я бачу, то й одержую (англ.)) HTML-редактором (наприклад, *FrontPage*) людину, яка до цього користувалася тільки текстовими редакторами й нічого не знає про HTML, вона цілковито точно очікуватиме, що її картинка буде збережена в самому документі, а не в окремому файлі. Можна назвати це *інерцією моделі користувача*.

У підсумку виходить конфлікт між моделлю користувача (картинка буде збережена в документі) і моделлю програми (картинка повинна бути збережена в окремому файлі). Дизайн інтерфейсу стане джерелом проблем.

Як вирішити таку проблему? Змінити HTML не вийде. Але треба придумати щось, що приведе модель програми у відповідність із моделлю користувача.

Є вибір. Можна спробувати змінити модель користувача. Це виявиться неймовірно складним. Можна пояснити все в керівництві, але всім відомо, що користувачі керівництва не читають, напевно, і не зобов'язані. Можна створити спливаюче діалогове вікно з повідомленням, що картинка в документі збережена не буде. Це викличе *дві* додаткові проблеми: по-перше, діалогові вікна дратують просунутих користувачів, по-друге, повідомлення у діалогових вікнах також ніхто не читає.

Тому практично завжди кращим рішенням буде змінити модель програми, а не модель користувача. Наприклад, при додаванні картинки її копія міститься в піддиректорію файлу, що відповідає уявленню користувача про те, що картинка копіюється (а оригінальний файл може бути вилучений).

Основні причини невідповідності концептуальної моделі моделі користувача:

- проектувальники думають, що знають аудиторію й конструюють інтерфейс у відповідності зі своїми припущеннями, не спромагаючись їх перевірити, і раптом виявляється, що їхні припущення невірні;
- змінюється цільова аудиторія продукту. Його створювали для певної людини або групи людей, і концептуальна модель відповідала ментальній моделі цієї групи;
- реальна робота над дизайном не була проведена. Концептуальна модель зовсім не створювалася. Вона являла собою лише відбиття апаратних засобів, програмного забезпечення або бази даних, так що єдина аудиторія, який відповідає цей продукт – програмісти-розробники.

Єдиний варіант пристосувати ментальну модель під концептуальну – почати навчання користувачів ще до того, як у них з'явиться програмний продукт. Найбільш ефективним способом навчання є навчальне відео, де не розповідають, а демонструють.

5.10 Засоби підтримки користувача

Незважаючи на те, що користувачі керівництв не читають, у найскладніших ситуаціях, коли нема в кого спитати, вони все ж звертаються до довідки. Тому одним з основних аспектів проектування користувацького інтерфейсу є *довідкова система*. Довідкову систему додатка становлять:

- повідомлення, які генерує система у відповідь на дії користувача (зворотній зв'язок);
- діалогова довідкова система;
- документація, що поставляється із системою.

При знайомстві з програмним продуктом у користувача можуть виникати питання такого типу:

1. Інформаційний: що я можу робити з цією програмою?
2. Описовий: що це таке? Що воно робить?
3. Стосовний до процесу: як я це роблю?
4. Інтерпретаційний: що відбувається? Чому це відбувається? Що це означає?
5. Навігаційний: де я перебуваю? Звідки я прийшов і куди йду?
6. Вибірний: що я можу зробити тепер?
7. Стосовний до рекомендацій: що я тепер повинен зробити?
8. Історичний: що я зробив?
9. Мотиваційний: чому я повинен використати саме цю програму? Як вона мені допоможе?
10. Інвестиційний: що ще я повинен зробити? Чи не упустив я чого-небудь?

На підставі цього можна виділити декілька факторів проектування довідкової системи:

1. Зміст: довідкова система повинна знати, що робить користувач, і реагувати на його дії повідомленнями відповідного змісту.

2. Професійний рівень користувача: повідомлення повинні містити відомості, які відповідають професійному рівню користувачів.

3. Досвід користувача: якщо користувачі добре знайомі із системою, їм не потрібні довгі й докладні повідомлення. У той же час починаючим користувачам такі повідомлення здадуться складними, мало зрозумілими й занадто короткими.

Перше враження користувача про систему засновано на повідомленнях про помилки. Вони повинні:

- бути послідовно конструктивними;
- бути ввічливими й короткими;
- не містити звукових сигналів, які можуть спантеличити відвідувачів.

Бажано:

- зв'язати повідомлення з контекстно-залежною довідкою;
- включити у повідомлення варіанти виправлення помилки.

Більш докладно рекомендації щодо повідомлень про помилки представлені в темі №3 про властивості інтерфейсу (дружність).

Контекстно-залежною (або просто контекстною) називається довідка, конкретний зміст якої визначається програмою автоматично, за сформованою на момент виклику поточною ситуацією.

Якщо, наприклад, у момент звертання до довідки користувач працює з операційним меню, то контекстно-залежна довідка буде містити інформацію про правила роботи з меню й опис конкретних пунктів цього меню. Якщо ж виконується будь-яка робота зі значками, то користувачеві буде надана довідкова інформація про значки, їхні типи або правилах роботи з ними. Якщо виведено повідомлення про помилку, то довідка має надати докладний опис цієї помилки й методів її усунення.

Крім контекстно-залежної довідки існують такі стандартні засоби як предметно-орієнтована допомога, довідник і майстер.

Проблемно-орієнтована допомога являє собою опис послідовності кроків, необхідних для виконання деякого завдання користувача. Для надання

користувачеві проблемно-орієнтованої допомоги відповідна довідкова інформація організується у вигляді розділів, кожний з яких описує окремий крок завдання. У свою чергу, кожний такий розділ відображається на екрані у вигляді окремого вікна, називаного вікном *Розділ завдання* (рис. 5.11).

Це вікно по можливості повинне займати мінімум простору основного вікна додатку, що дозволить користувачеві прочитати виведену інформацію, не переміщаючи вікно. Рекомендується використовувати для основних розділів такий розмір вікна, щоб для перегляду інформації не були потрібні смуги прокручування.

Заголовок вікна Розділ завдання повинен однозначно ідентифікувати відображувану в ньому інформацію.

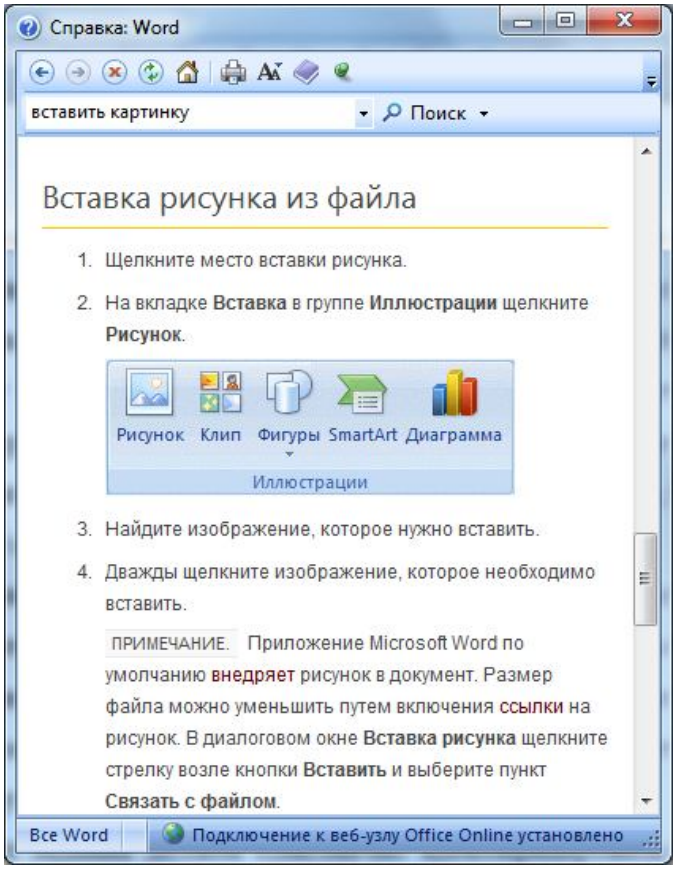


Рис. 5.11 Предметно-орієнтована допомога

На відміну від контекстно-залежної підказки, довідкова інформація з розділів завдання повинна бути сформульована у вигляді відповіді на питання «Яким чином?», а не «Що?» або «Чому?», оскільки покликана допомогти

користувачеві у виконанні конкретного завдання, а не просто розширити його знання з відповідної теми.

Довідник забезпечує надання користувачеві довідкової інформації у формі інтерактивної документації. Використання довідника допомагає користувачу усвідомити загальні основні характеристики програмного продукту.

Доступ до довідника може бути реалізований декількома способами. Найпоширеніший з них – явний виклик за допомогою відповідної команди *Довідка* з меню, що випадає, але можливе також використання спеціальної кнопки на панелі інструментів або виклик через піктограму конкретного об'єкту.

Хоча довідник може містити інформацію подібну тієй, котра виводиться у вікнах контекстної підказки й проблемно-орієнтованої допомоги, ці форми допомоги не виключають один одного.

Кожне вікно (сторінка) довідника може включати текст, графіку, елементи анімації, відеокліпи й звуковий супровід.

Майстер являє собою спеціальну форму допомоги користувачеві, що дозволяє автоматизувати виконання завдання за допомогою ведення діалогу. Майстри використовуються в тих випадках, коли виконуване завдання є досить складним і вимагає значного досвіду в роботі з додатком. Діапазон застосування майстрів досить широкий. За допомогою них може бути автоматизоване практично будь-яке завдання, включаючи створення нових об'єктів (наприклад, побудова графіка) або форматування вже існуючих (наприклад, таблиці або параграфа). Але виконання таких завдань майстром можна скоріше ускладнити, ніж спростити. Частіше майстри використовуються для установки нового обладнання або програмного забезпечення (рис. 5.12).

Майстрів не слід використовувати для навчання користувача: хоча вони й допомагають користувачеві у виконанні завдання, майстри повинні розроблятися таким чином, щоб сховати від нього багато кроків завдання й зовні спростити завдання. Майстер повинен оперувати з реальними даними. Не варто також розраховувати на те, що застосування майстрів дозволить

підвищити якість непродуманого або складного інтерфейсу додатку. Більш за те, майстрів варто розглядати лише як доповнення до основних інструментів додатку, призначеним для виконання конкретного завдання.

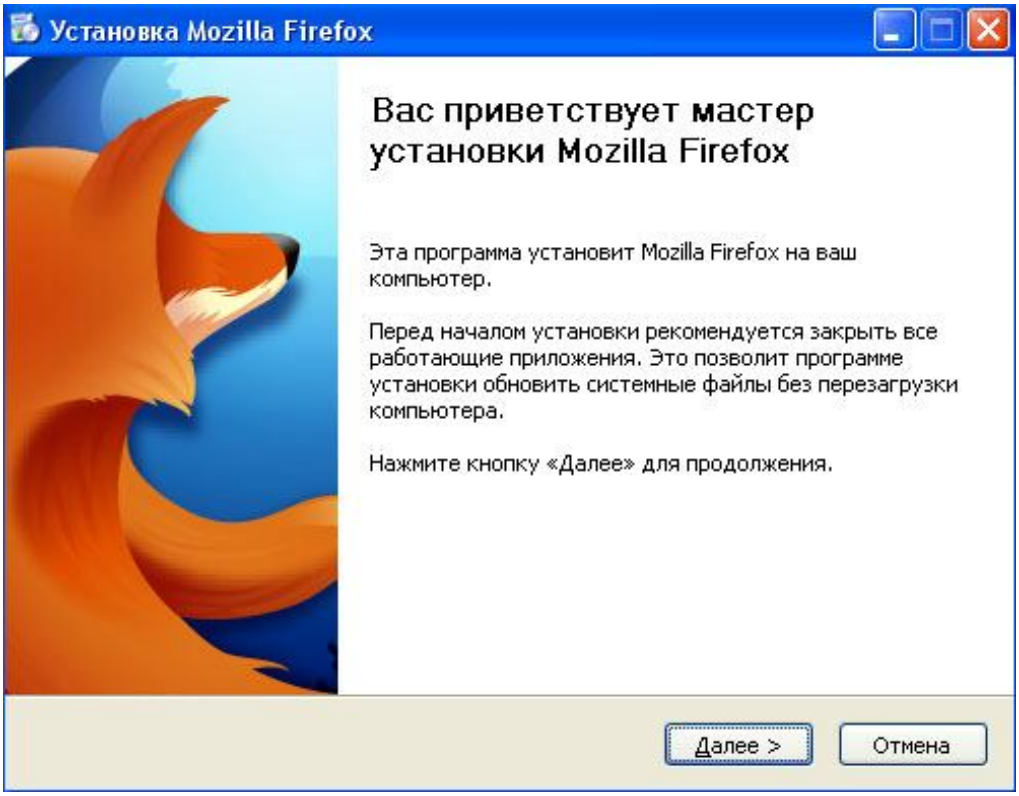


Рис. 5.12 Майстер установки програмного забезпечення

У зв'язку з тим, що довідкова система має ієрархічну структуру, де на верхніх рівнях ієрархії міститься більш повна інформація, а на нижніх – більш докладна, може виникнути наступна ситуація: користувач заходить у систему, одержавши повідомлення про помилку, і потім переміщується в системі за посиланнями. Через якийсь час він заплутається і йому необхідно починати все з початку. Щоб таких ситуацій не виникало інформацію зручно відображати в декількох вікнах (рис. 5.13).

Тексти довідкової системи повинні бути:

- написані разом із розробниками додатку;
- продумані так, щоб їх можна було прочитати у вікні малого розміру (тільки необхідна інформація);
- адаптовані до недосвідченого користувача.

Документація користувача повинна містити 5 документів:

- функціональний опис, у якому коротко представлені функціональні можливості системи. Прочитавши функціональний опис, користувач повинен визначити, чи та це система, яка йому потрібна;
- документ про інсталяцію системи, у якому міститься інформація з установки системи;
- вступне керівництво, яке представляє неформальне введення в систему, яке описує її "повсякденне" використання;
- довідкове керівництво, у якому описані можливості системи й їхнє використання, представлений список повідомлень про помилки й можливі причини їхньої появи, розглянуті способи відновлення системи після виявлення помилок;
- керівництво адміністратора, необхідне для деяких типів програмних систем. У ньому надається опис повідомлень, які генеруються системою при взаємодії з іншими системами, й описані способи реагування на ці повідомлення.

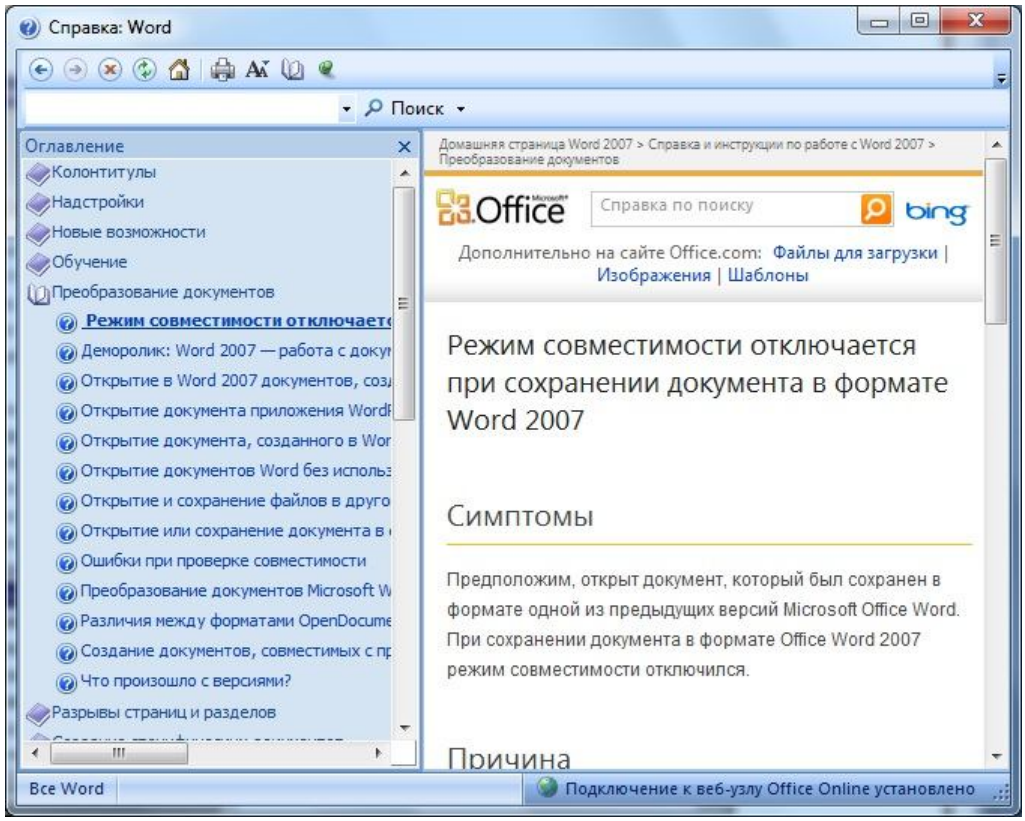


Рис. 5.13 Вікно довідки

Разом з перерахованими керівництвами необхідно надавати іншу зручну в роботі документацію. Для досвідчених користувачів системи зручні різного виду предметні покажчики, які допомагають швидко переглянути список можливостей системи й способи їхнього використання.

5.11 Що мотивує людину

При використанні програмних продуктів у робочих цілях, особливо якщо цей продукт нав'язаний самим підприємством, людину головним чином мотивує його зарплатня. Тому залучення нових користувачів часто більше залежить від роботи відділу маркетингу, чим від властивостей самого продукту. У сфері веб-додатків, в ігровій індустрії ситуація прямо протилежна. Тут іде боротьба за залучення й утримання уваги кожного окремого користувача. Тому важливо знати, як мотивувати людину користуватися конкретним продуктом.

Мотивація підсилюється при наближенні до мети. Чим менше відстань до мети, тим більше люди мотивовані досягти її. Мотивація стає ще сильніше, коли мету видно. Так у часи великих географічних відкриттів моряки в далеких плаваннях засмучувалися, коли довго не могти досягти берега, і скільки випадків знає історія, коли галас вахтового «Земля, земля!» міг підняти знесилених штормами, відсутністю їжі й води людей і направити судно до цілі.

Можна досягти цієї додаткової мотивації навіть за допомогою *ілюзії прогресу*. Зараз багато супермаркетів пропонують збирати фішки. Фішки видаються за умови покупки товарів у супермаркеті на певну суму (звичайно від 20 до 50 гривень). Зібравши певну кількість фішок, покупець зможе придбати деякий «ексклюзивний товар» зі значною знижкою. Наприклад, зібравши 10 фішок можна купити каструльку, яка «самоварить», із знижкою 50%. Покупцеві видається спеціальна картка із зображенням осередків, куди треба вклеювати фішки. Порожня картка наводить на міркування, а чи точно потрібна така каструлька, але коли осередки починають заповнюватися, каструлька починає виглядати все привабливіше. Якщо видати картку, у якій для одержання тієї ж каструльки потрібно буде зібрати 12 фішек, але дві вже будуть уклеєні як подарунок від магазина, то все буде виглядати вже так,

начебто деякий рух уже є, й ефект той же самий, як у випадку реального просування. Цей ефект досліджував ще в 1934 році Кларк Хулл.

Міньюнг Ку й Айелет Фішбач в 2010 році проводили дослідження з метою визначення, що більше мотивує людей до досягнення мети – те, що вже зроблено або те, скільки залишилося. Була обрана друга відповідь.

На веб-сайтах ці принципи часто використовують, щоб мотивувати людей до активного поводження. Крім того, було виявлено, що людям подобається брати участь у програмах з нагородами.

Наприклад, пропонують нагороду, часто віртуальну, якщо користувач оцінить певну кількість пісень, завантажить деяку кількість фотографій і т. п. Сайт Dropbox, який надає користувачам «жорсткий диск в Інтернеті», пропонує додаткове місце для зберігання даних за залучення нового клієнта. При цьому, щоб показати як близько людина підійшла до бонусу, шлях до його одержання представляється у вигляді шести кроків, серед яких установка самого додатку на комп'ютер, запис файлів користувача на основну частину дискового простору. Тому зробивши те, що користувач і так би зробив, він бачить, як на шляху одержання бонусу вже виконано більшу частину кроків, і залишилося один або два кроки, щоб досягти мети.

Шкали прогресу часто застосовуються на сайтах з онлайн курсами. Так наприклад на рис. 5.14 показано сторінку з контрольними тестами сайту, присвяченому курсу Стенфордського університету з комп'ютерних мереж. Напроти кожного тесту розташована смуга прогресу виконання. Якщо вона темна – тест пройдений повністю, світліша – показує пройдену частину тесту, а біла – тести, яких студент не торкався.

Так само на сайті є кілька вбудованих форм, які показують прогрес навчання повністю. Користувач відразу може визначити в якому місці курсу або уроку перебуває й чого вже досяг.

Мотивація падає відразу після того, як ціль досягнута. Це явище називається відновленням після одержання винагороди. Навіть якщо запропонувати наступний рівень, люди будуть не занадто прихильні одержати

другу нагороду. Тому найбільший ризик втратити клієнта виникає відразу після досягнення мети й одержання нагороди. Управляти цим ризиком можна за рахунок частоти винагород і відповідністю самої винагороди потребам аудиторії.

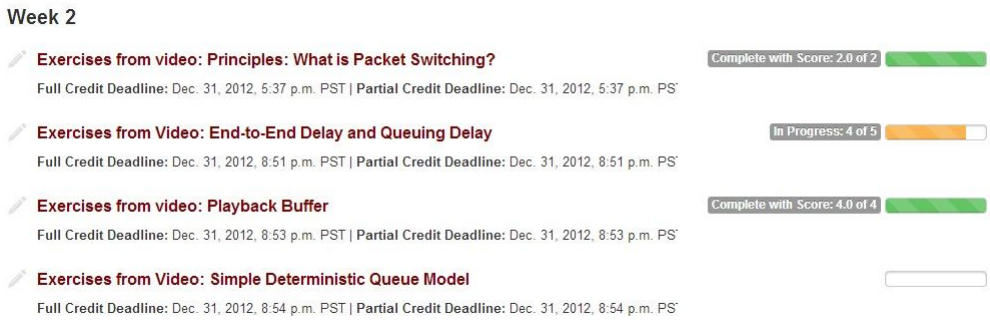


Рис. 5.14 Демонстрація наближення студента до мети на сайті class.stanford.edu/networking/

На згаданому раніше сайті Dropbox людина одержує додаткове місце для зберігання за кожного приведеного клієнта на сайт. Такий спосіб називається постійною схемою підкріплення. Американський психолог Б. Ф. Скіннер, автор оперантного навчання й ініціатор навчання за допомогою спеціальних машин, запропонував наступні *схеми заохочення*:

- засновані на інтервалі;
- засновані на кількості дій.

Кожна схема може бути фіксованою або змінюваною. Тобто існує 4 можливих варіанти:

1. Фіксований інтервал. Підкріплення залежить від часу. Інтервал часу завжди той самий;
2. Змінний інтервал. Підкріплення залежить від часу. Інтервал змінюється, але середнє значення зберігається;
3. Фіксована кількість дій. Підкріплення засноване на кількості дій, і це число завжди однаково;
4. Змінювана кількість дій. Підкріплення засноване на кількості дій. Кількість щоразу міняється, але середнє значення зберігається.

У результаті проведених Скіннером досліджень виявилося, що варіант кожної схеми зі змінюваним параметром завжди дає кращий відгук від людини. Цим користуються всі казино.

Але в цілому схема, заснована на числі дій, показала себе краще. Таким чином, Dropbox міг би досягти кращого результату, якби збільшував нагороду залежно від кількості наведених друзів. Тобто перейшов від постійної схеми до схеми з фіксованим числом дій.

Ці принципи заохочень важливо використовувати при розробці стратегії ігор. Якщо користувач у ході тривалого часу або великої кількості спроб не може досягти позитивного результату, то він просто закине таку гру. Тому його потрібно заохочувати через певні інтервали.

Зовнішня винагорода. Користувач зовсім не обов'язково повинен одержувати якийсь відчутний подарунок. Звичайно гроші й інші коштовні призи це серйозний стимул, але не завжди кращий. Після одержання коштовної винагороди люди будуть орієнтуватися тільки на нього й не стануть що-небудь робити, якщо немає матеріальної мотивації. Тому найбільше ефективна призова винагорода, якщо вона несподівана.

Винагорода, яка дається за певне поводження, запропоноване протягом деякого часу, дає менший результат у вигляді бажаного поводження, якщо нагорода не повторюється.

З метою мотивації найкраще натискати на *внутрішні стимули* й психофізіологічні особливості людини. Така мотивація більше довгострокова. Існує безліч жартів на тему того, як Інтернет затягує людину, змушуючи забути про роботу, прибирання квартири, їжу й багато інших речей. Коли, здавалося, присів на хвилиночку переглянути пошту, а схаменувся через кілька годин на «іншій стороні» Google. Спрага дослідження й одержання задоволення від відкриття не дає людині залишити непрочитаним повідомлення, яке прийшло на електронну адресу. Ця властивість мозку дозволила людству розвиватися. Якби люди не були стурбовані пошуком нових предметів й ідей, то дотепер сиділи в печерах. В 1958 році нейробіологи зі Швеції ідентифікували

дофамінову систему – сукупність органів, які виробляють дофамін. Гормон дофамін необхідний для всіх функцій мозку, включаючи мислення, рух, сон, настрій, увагу, мотивацію, пошук і винагороду. Саме дофамін змушує людину шукати інформацію. Дофамін виділяється в передчутті чого-небудь і відразу після досягнення мети. Він визначає здатність насолоджуватися життям. Виходить, що передчуття, краще володіння.

Передчуття знайденого приводить до так званому веб-серфінгу, коли користувач почав шукати одну інформацію, а через півгодини вже переходить за посиланнями зовсім в іншій темі. Чим простіше пошук, тим більше людина затягується в цей процес. Шукати інформацію в інтернеті простіше, ніж шукати корисні копалини, шукати ту ж інформацію в бібліотеці або навіть у книзі, яка перебуває під рукою.

Стан пошуку підтримується ефектом несподіванки, тому що несподіванка стимулює виробіток дофаміну. Що змушує людину переглядати пошту, хоча вона тільки п’ять хвилин назад заглядала у поштову скриньку? Саме те, що момент приходу нового листа він точно не знає. Виділення дофаміну дуже відчутно до сигналів, які показують, що наближається винагорода. Якщо надходить сигнал, який вказує на деяку подію, то спрацьовує той самий рефлекс собаки Павлова. Програми для онлайн спілкування, наприклад Skype (рис. 5.15), і соціальні мережі не даремно сповіщають звуковим і візуальним сигналом про будь-який рух – появу в мережі нового друга, оцінювання фотографії, зміну статусу й т. п. Павловський рефлекс відразу змушує людину шукати інформацію.

Рис. 5.15 Візуальний сигнал про нове повідомлення

Як уже згадувалося, людина любить шукати, але не любить при цьому робити значні зусилля. Дофамінова система стимулюється найбільш енергійно, коли інформація надходить у малих порціях. Це один із секретів популярності сервісу (служби) для ведення мікроблогів Twitter – велика кількість

користувачів, від яких постійно приходять нові повідомлення, довжина яких всього 140 символів. Електронна пошта ставить за обов'язок людині скласти лист, а лист це помітний блок інформації. В той час соціальні мережі пропонують писати коментарі й повідомлення на стіну – кожному зрозуміло, що на стіні багато не напишеш. У популярності соціальних мереж переконувати не доводиться.

Внутрішні стимули. Існує багато видів діяльності, якими із задоволенням займаються люди, хоча ці види діяльності вимагають великої кількості часу, знань і не приносять ні фінансової вигоди, ні навіть кар'єрного росту.

Рис. 5.16 «В Контакте» показує прогрес у заповненні профілю

Людям подобається відчувати, що вони сприяють прогресу, що вони вчаться й створюють нові знання й навички. Саме тому люди віддають свій час наповненню Вікіпедії й інших відкритих джерел. Тут у приклад можна привести й раніше згадану шкалу прогресу, яка показує, скільки вже зроблено й скільки залишилося до «досконалості».

Так у соціальних мережах часто спонукають закінчити заповнення профілю, показуючи, скільки вже інформації користувач видав (рис. 5.16).

У додатках, де складна дія розбивається на невеликі кроки, наприклад у майстрах установки програм, бажано відразу показувати загальну кількість кроків, і скільки вже пройдено. На рис. 5.17 показано, що в інтернет-магазині Bonprix процес замовлення товару складається із чотирьох кроків, пояснене призначення кожного кроку, показано на якому кроці зараз перебуває користувач і стан виконання кроку. Судячи із зображеного на рисунку, користувач виконав перший крок і може перейти до другого.

У додатках для роботи людина обов'язково повинна закінчити почату задачу, тому що її підтискає час, начальство й інші зовнішні фактори. Відсутність кінцевого результату її буде тільки дратувати. І тут залучити клієнта надовго можна за рахунок його повного задоволення. В іграх навпроти, щоб утримувати постійний інтерес можна скористатися неможливістю досягнення досконалості. Можна ставати усе краще й краще, але ніколи не досягти кінцевої точки. Саме це робить досконаліть непереборним мотивуючим фактором.

Рис. 5.17 Інтернет-магазин Bonprix - покрокове здійснення замовлення

Людей мотивує можливість установлення зв'язків. Людина істота соціальна. Вона може побажати використати продукт просто тому, що це дозволить контактувати їх з іншими людьми.

При створенні продукту, призначеного для побудови або використання соціальних зв'язків, варто визначити які саме це повинні бути зв'язки – сильні або слабкі. Відносини з великою кількістю людей через соціальні мережі можна вважати слабкими зв'язками. Стійкі соціальні відносини можливі в групі людей від 100 до 230 чоловік, не більше. *Стійкі соціальні відносини* – це відносини, у яких відомо, ким є кожний учасник групи і як він ставиться до інших учасників групи. Встановлення таких зв'язків припускає можливість фізичного контакту й такої взаємодії в мережі, щоб люди могли допитатися один одного.

Багато видів взаємодії в мережі не є спільними. Хоча така соціальна діяльність характерна для нашого повсякденного життя, вона не може

заповнити бажання людини брати участь у спільній діяльності й одержувати від неї особливе задоволення.

Взаємодії в мережі підкоряються правилам особистої взаємодії. Тому правила розробки споживчих властивостей такого продукту є правилами, пов'язаними із прийнятими в суспільстві очікуваннями для взаємодії.

Якщо веб-сайт не відповідає або занадто довго завантажується, це виглядає так само, якби людина, до якої ви звертаєтеся ігнорувала вас. Якщо веб-сайт не зберігає інформацію від сесії до сесії, це схоже на те, що співрозмовник вас ігнорує, не впізнає або робить вигляд, що ви не знайомі.

Потрібно запропонувати людям діяльність, яка подобається їм за своєю природою (спілкування із друзями, створення чогось нового), а не просто діяльність, за яку вони одержують оплату. Якщо люди вирішують нудне завдання можна мотивувати їх, визнавши, що завдання нудне, і дозволивши вирішувати його власними методами. Це допоможе забезпечити постійних клієнтів.

Самостійність як засіб мотивації. У темі про властивості інтерфейсу говорилося, що хороший інтерфейс такий, коли людина відчуває, що вона управляє програмою, а не програма управляє нею. Чому популярні супермаркети? Тому що там не треба чекати поки прийде продавець, можливо в поганому настрої, й покаже товар – можна подивитися все самому. Інтернет-магазини рятують навіть від контакту з незадоволеними касирами й не обмежують відвідувачів ні годинами роботи, ні кількістю проведеного у магазині часу, ні кількістю спроб роздивитися товар. Термінали самообслуговування, онлайн-банкінги – всі ці продукти дозволяють самостійно робити деякі речі, замість того, щоб діяти через інших людей. Звичайно вони повинні бути дуже зрозумілими, щоб не викликати почуття страху, яке переважить бажання бути самостійним.

Самостійність мотивує людей, оскільки дає їм почуття контролю. Підсвідомості подобається, коли все під контролем. Наприклад, користувачі можуть віддати перевагу користуванню шаблоном створення сайтів, замість

того щоб наймати фахівця й пояснювати йому що та як потрібно, а потім одержати продукт, який не буде відповідати їхнім очікуванням, ще й заплатити за це гроші. Прикладами таких додатків може бути narod.ru, ucoz.ua (рис. 5.18). На головній сторінці сайтів відразу пояснюються користувачеві його можливості.

Рис. 5.18 Ucoz дозволяє створювати сайти самостійно

Тема 6. **Функціональні компоненти**

6.1. Популярні стилі користувацького інтерфейсу

По суті, всі популярні сьогодні стилі можна віднести до *графічного користувацького інтерфейсу* (або GUI – Graphical User Interface). Визначається він як стиль взаємодії «користувач-комп'ютер», у якому застосовуються чотири базових елементи: вікна, піктограми, меню й вказівники. Іноді GUI-інтерфейс називають WIMP-інтерфейсом (Windows – вікна, Icons – піктограми, Menus – меню, Pointers – покажчики). Найважливіші властивості графічного інтерфейсу – це можливість безпосереднього маніпулювання, підтримка миші або вказівника, використання графіки й наявність області для функцій і даних додатку. На основі графічних інтерфейсів побудовані *Web-інтерфейси* (WUI –

Web User Interface). Стильові деталі останніх незначно відрізняються від графічних, проте все ж мають свої особливості, наприклад у наборі доступних функціональних елементів й більшої вільності у використанні візуальних ефектів, як то кольори, шрифти, картинки й т. п. Фактори успішного проектування web-інтерфейсів: простота навігації по ієрархічних інформаційних структурах, легкість і швидкість пошуку, швидка реакція. До інших важливих факторів відносяться естетичні характеристики й цінність поточного змісту інформації.

Ще одним напрямком розвитку графічного інтерфейсу став об’єктно-орієнтований стиль. *Об’єктно-орієнтований інтерфейс* прикладного користувальницького додатку повинен забезпечувати:

- безпосереднє маніпулювання (перетаскувати будь-які об'єкти куди завгодно);
- безпосереднє уведення даних (записувати будь-яку інформацію);
- контекстну залежність від об'єктів (спливаючі контекстні меню, довідки, погодженість і т. п.).

6.2. Первинні й вторинні вікна

Вікно є спеціальна область фізичного екрану, за допомогою якої користувач має можливість одержати відображення певного аспекту розв'язуваного завдання.

Поняття *первинного* вікна. Цей термін уведений для позначення вікон, іменованих в англомовній літературі *primary window.* Вікна такого типу звичайно використовуються для подання вмісту об'єктів або в якості основної робочої області додатку. У слов’янськомовних (у тому числі перевідних) виданнях вони або взагалі не мають спеціального позначення (просто *вікна),* або йменуються *головними* вікнами.

Ознаки первинного вікна:

- рядок меню;
- панель інструментів;

- смуги прокручування;
- рядок станів.

Типи *вторинних* вікон:

- діалогове вікно (відкрити, закрити, зберегти, авторизуватися);
- папка із вкладками (властивості, параметри, настроювання, коли обсяг інформації перевищує корисну площу);
- список, що випадає;
- вікно повідомлень.

Типові кнопки вторинних вікон: так, відміна, допомога.

Правила проектування роботи вікон:

1. Коли користувач відкриває нове вікно, воно повинне відображатися поверх інших вікон того ж рівня й установлюватися в активний стан (як правило, всі первинні вікна ставляться до одного рівня). Додаткові, або *вторинні* вікна, які відносяться до даного додатку, також повинні відображатися поверх інших вторинних вікон того ж рівня.

2. Якщо користувач, запустивши додаток, встиг повернутися до роботи з іншим вікном раніше, ніж відкрилося нове вікно, то при відкритті нове вікно не повинне «заслонити» активне вікно.

3. Повторне виконання команди, за допомогою якої було відкрите вікно, повинне активізувати існуюче вікно замість відкриття іншого екземпляра вікна. Необхідно мати на увазі, що наведене правило ставиться до локального Робочого стола користувача. Якщо ж два користувачі, які працюють у мережі, відкривають вікно для того самого мережевого об'єкта, то кожний з них може бачити вікно цього об'єкта на своєму Робочому столі.

4. Закриття вікна не обов'язково означає завершення процесів, пов'язаних з об'єктом, який представлений у вікні. Наприклад, закриття вікна принтера не скасовує печатання документів, які очікують своєї черги. Вихід з додатку завжди приводить до закриття всіх його вікон, але закриття якого-небудь вікна не обов'язково приводить до виходу з додатку. Саме тому у вторинних вікнах

не рекомендується використовувати слово «Закрити» для позначення кнопок, пов'язаних із закриттям вікна (але не додатка!). Більш відповідними в цьому випадку є назви «Застосувати», «Добре», «Скасувати» і т. п. Не рекомендується також використовувати команду Закрити як еквівалент команди Відмінити. Побічний ефект закриття вікна за допомогою команди Закрити залежить від поточної ситуації.

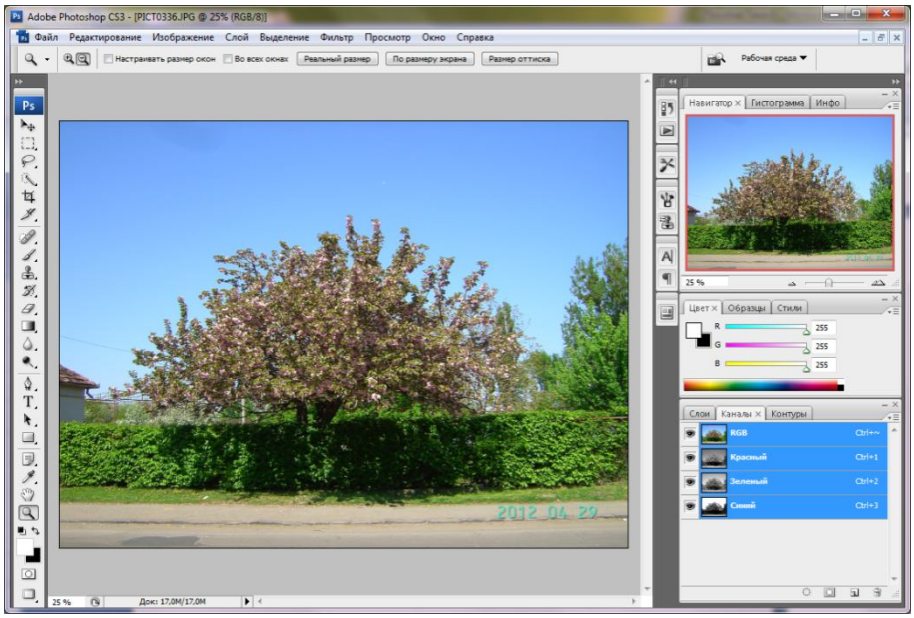


Рис. 6.1 Використання панелей у графічному редакторі

У теперішній час тенденції спрямовані убік переваги *панелей* перед вторинними вікнами, якщо справа стосується властивостей, параметрів або настроювань (рис. 6.1). Розміри сучасних моніторів дозволяють перебувати їм постійно на екрані, користувач не витрачає час на пошук цих даних у меню, відкриття й закриття вікон. До того ж панелі не перекривають робочу область.

Панелі дозволяють більш ефективно використати передпросмотр і контролювати результат внесених змін.

У графічних інтерфейсах, особливо в Windows, поширене поняття документ.

Документ – це механізм, який дозволяє зібрати інформацію в додатку таким чином, щоб користувач міг взаємодіяти з нею.

Документ може містити як текстові, так і інші групи даних.

Звичайно на екрані відображається тільки фрагмент даних документа. Цей фрагмент називається його *поданням*.

Простий додаток може складатися з одного вікна, у якому відкривається тільки один документ, наприклад Paint. Такий інтерфейс називається SDI (*single document interface*) – однодокументний.

Для складних додатків може знадобитися одночасна робота з декількома документами, як однотипними, так і різного типу. Такий інтерфейс називається багатодокументним (MDI – *multi document interface*).

MDI-додатки використовують одне головне вікно, яке називається *батьківським*. Кожний документ відкривається в окремому вікні, котре називається *дочірнім*. На логічному рівні дочірні вікна поводяться так, ніби кожне з них є головним.

Усі дочірні вікна використовують один рядок станів, меню, панель інструментів.

6.3. Піктограми

Піктограми роблять інтерфейс більше привабливим у візуальному відношенні й, за певних умов, можуть сприяти більшій зрозумілості. Однак згодом стали виразними й недоліки піктограм. Наприклад, як в операційній системі Macintosh, так і в Windows зараз уже використаються засоби для пояснення значення піктограм. Якщо користувач наводить курсор на якусь піктограму, з'являється невелике вікно з текстом, у якому дається її опис. Виникає очевидне питання: «Чому замість піктограм відразу не використати текст?» Проблему піктограм можна розглядати як проблему обмеженої видимості. В інтерфейсі показана піктограма, але її зміст не бачимий, або ж її зображення може дати невірне повідомлення для тих, для кого це зображення незнайоме або ким це зображення може бути витлумачене по-іншому. Наприклад, піктограма, яка зображує долоню піднятої руки, у Сполучених Штатах означає «стоп», а у Греції - «от вам екскременти на вашу особу».

Відповідно до проведених досліджень, піктограми є найбільш ефективними, якщо кількість піктограм, які видно одночасно, не перевищує 12.

Необхідні властивості піктограм:

- візуально відрізнятися одна від одної;
- добре відображати відповідне поняття;
- мати розумно великий розмір (звичайно більший, ніж міг би бути текстовий надпис).

Як це не дивно, піктограми порушують принцип видимості, оскільки саме зміст піктограм залишається невидимим.

Звичайно, слова не завжди виявляються підходящим засобом. Наприклад, це стосується колірних палітр. У деяких програмах палітра передається за допомогою слів, представлених за абеткою. Ось частина цього списку: Bittersweet (пасльоновий), Burnt Orange (палений жовтогарячий), Burnt Sienna (палена охра), Corn Flower (волошковий), Lemon Yellow (лимонний), Goldenrod (золотушниковий).

Який колір означає назва пасльоновий? А що робити тим, хто не знаком з ботанікою й не може відрізнити ромашку від волошки? Тобто застосовувати піктограми потрібно тільки в тих випадках, коли дослідження показують, що їхнє застосування може бути більше ефективним. В інших випадках слова краще.

Виділення кольорами може бути марним, якщо використовується занадто багато кольорів або, якщо є занадто багато піктограм одного кольору. У випадку, коли піктограми розташовані близько одна від одної, не слід використовувати більше семи кольорів; якщо ж піктограми розкидані по всьому екрані, то число кольорів не повинне перевищувати чотирьох.

6.4. Меню

Меню містить перелік команд, наявних у розпорядженні користувача при виконанні певного кроку завдання або завдання в цілому. Меню надає

користувачеві можливість вибору необхідного засобу рішення завдання, не вимагаючи від нього запам'ятовування імен команд й їхнього синтаксису.

Одна з найпоширеніших форм меню – *лінійна* послідовність команд (або розділів). Саме в такому виді виконане *головне меню* вікна, розташоване безпосередньо під смугою заголовку первинного вікна. У зв'язку із цим головне меню називають також *смугою меню.* Смуга меню містить назви пунктів меню, кожний з яких надає доступ до меню, що випадає. Зміст головного меню й пов'язаних з ним меню, що випадають, визначається функціональним призначенням додатку й контекстом виконуваного користувачем завдання. Головне достоїнство меню, що випадає, – економія місця. Але при цьому воно має значні недоліки: необхідність повного перегляду, ускладненість перегляду, нестійкість сприйняття. У додатках з великою кількістю функцій ці недоліки збільшуються каскадним меню, коли пункти одного меню, що випадає, викликають інше меню, що випадає (рис. 6.2.).

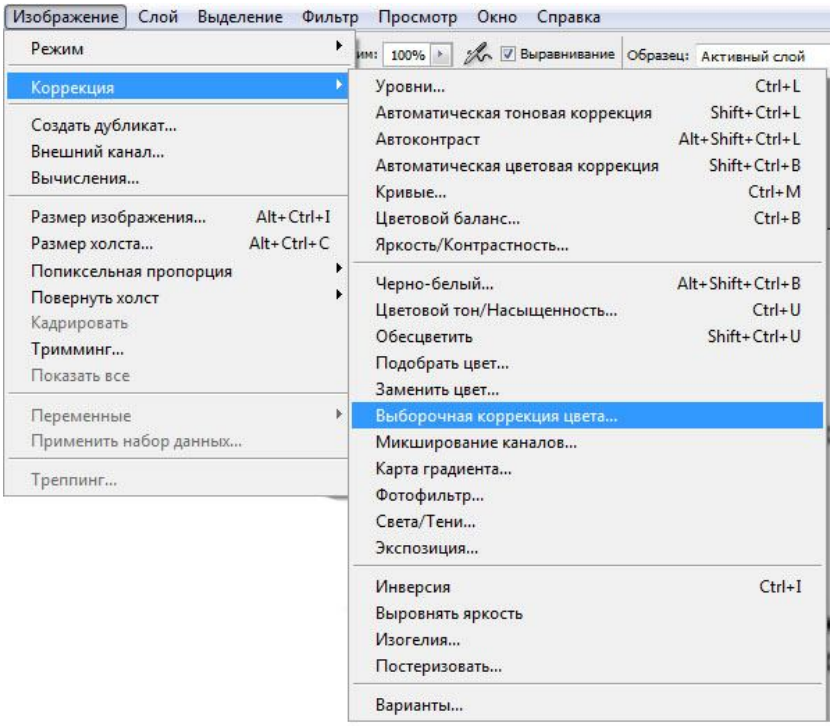


Рис. 6.2 Лінійне, що випадає й каскадне меню

Тому в Microsoft Word 2007 перейшли від каскадного меню до меню типу стрічка, з використанням вкладок (рис. 6.3.).

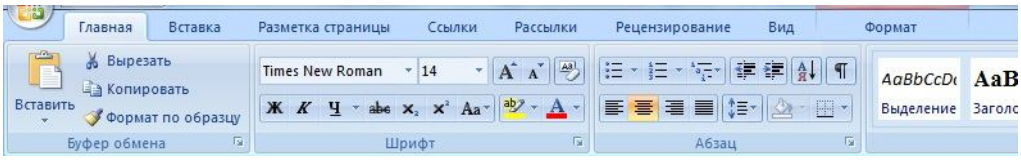


Рис. 6.3 Меню-стрічка з використанням вкладок

Вкладки визнані однією з найбільш удалих метафор у якості засобу навігації, тому що вони самоочевидні й приємні для очей, створюють ефект фізичного простору, їх важко не помітити. При використанні вкладок треба дотримуватися декількох правил:

- вкладки повинні бути чітко прорисовані;
- вкладки повинні виділятися кольорами;
- одна із вкладок повинна бути обрана.

Ще один варіант меню типу стрічка – *панель інструментів,* яка з'явилася в 1990 році. Microsoft Excel 3.0 був першим додатком, у якому застосовано панель інструментів. Річ виявилася потрібною, всім сподобалася, всі розробники її скопіювали. Додаток без панелі інструментів зараз явище досить рідке. Панель інструментів повинна містити піктограми команд, які найчастіше застосовуються. Тому, якщо панелей дуже багато, то губиться зміст їхнього використання.

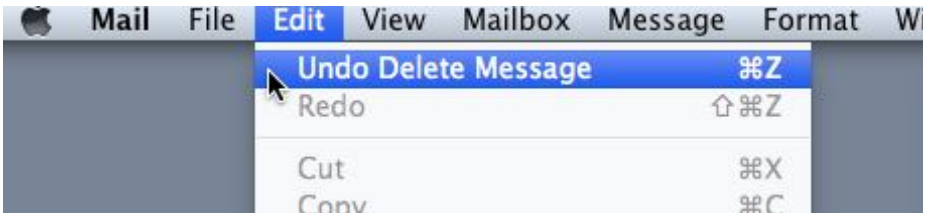


Рис. 6.4 Інтелектуальні пункти меню

Мітки меню варто міняти динамічно, точно показуючи, що відбудеться при їхній активації. Стандартна функція Undo (Відмінити) повинна чітко показувати яку дію буде відмінено (рис. 6.4). Це робить *інтерфейс самоописовим*. Користувач не повинен зупинятися й думати, на який об'єкт буде зроблений вплив. Ще менше йому сподобається зробити випадково те, що не збирався, наприклад видалити Главу 8 замість замітки 3.

Панель дій (панель завдань) – група функцій, які відносяться до даного контексту, тобто можуть бути застосовані до кожного видимого елементу.

Використання панелі дій замість меню робить інтерфейс більше організованим і функції завжди доступними.

Рис. 6.5 Панель дій

Панель дій – це лінійне меню в плоскому поданні (рис. 6.5). Ще одна причина використати панель дій: якщо функції повинні бути згруповані, й ці групи не укладаються в стандартні пункти головного меню: Файл, Редагування, Вид, Інструменти.

Панель дій вимагає багато екранного місця, тому вона не підходить для маленьких пристроїв.

Рис. 6.6 Список, що випадає, зі смугою прокручування

Меню, що випадають (рис. 6.6) найбільше підходять для алфавітних списків відомих назв, наприклад країн, областей, товарів, оскільки їхнє читання не вимагає додаткового міркування.

Проте списки істотно менш зручні в тих випадках, коли потрібно вибрати щось, назву чого невідомо, особливо, коли елементи списку не розташовані за абеткою або список настільки довгий, що його потрібно прокручувати.

Список, що випадає, зручний, якщо відразу видно його межі (рис. 6.7), але незручно якщо його довжина більше 20 найменувань.

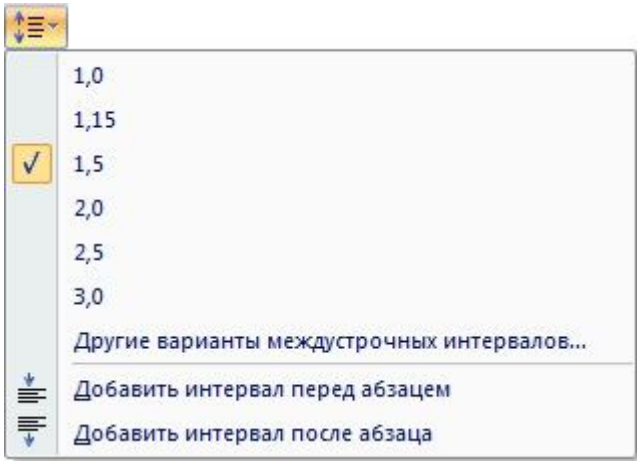


Рис. 6.7 Приклад зручного списку, що випадає

Смуга прокручування ліворуч (рис. 6.6) один з найбільш незручних елементів керування.

Скільки точно вивірених маніпуляцій мишею потрібно зробити, щоб у списку шрифтів вибрати Times New Roman, наприклад. Спочатку потрібно клацнути мишею по стрілочці, яка вказує вниз. Потім, акуратно переміщаючи квадратик керування смуги прокручування, переглядати список доти, поки не з'явиться потрібний шрифт. Багато подібних списків, що випадають, бездумно розроблені таким чином, що показують лише два або три елементи списку. Тому прокручування перетворюється в досить кропітку процедуру, особливо коли в списку десятки шрифтів. Коли вдалося добратися до *Times New Roman*, потрібно клацнути ще й по ньому. І якщо користувач промахнувся, прийдеться повторювати все спочатку.

Список, що випадає, потрібно робити таким довгим, щоб він уміщав всі елементи. Якщо місця між самим елементом управління й низом екрана недостатньо, щоб умістити всі елементи списку, треба використати простір до верху екрану. У випадку, коли елементів більше, ніж список у висоту екрана може вмістити, рекомендується встановити на ньому автоматичне

прокручування. Більше того, згідно законів Фітса й Хіка, краще для відкриття списку, що випадає, використовувати не маленьку стрілочку в куточку, а весь простір віконця. Це збільшує мету раз у десять і полегшує позиціювання вказівника миші у вікні.

Дурним тоном вважається використання горизонтальної смуги прокручування скрізь, крім великих таблиць, розкладів, графіків. Якщо збільшується розмір картинки, то смуги прокручування або інші позначення, які ілюструють, що малюнок більший за видиму область й є можливість переглянути інші частини, повинні з'являтися автоматично в необхідних напрямках.

Ще одним недоліком списку, що випадає, є вибір елементу, який приводить до зміни основного вмісту документа, є те, що він цей вміст перекриває. І зміни можна буде побачити тільки після закриття списку. Добре рішення цієї проблеми можна побачити в ранніх версіях Microsoft Word. Заголовок стилю відразу дає зрозуміти, як він виглядає (рис. 6.8).

Рис. 6.8. Приклад самоописового списку, що випадає

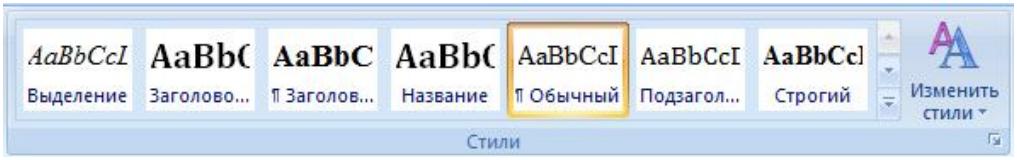
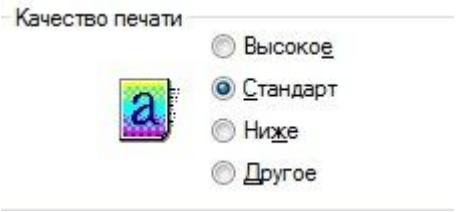


Рис. 6.9. Використання стрічкового меню замість списку, що випадає

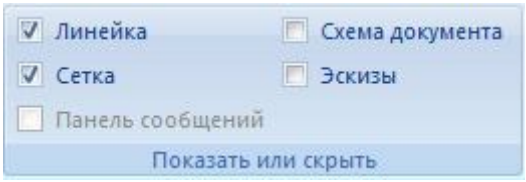
Проте у Word 2007 залишили цю перевагу, але від списку, що випадає, відмовилися (рис. 6.9).

Якщо список невеликий, то існує великий вибір функціональних елементів для його обробки:

- *радіокнопки* – вибір одного із запропонованих варіантів:

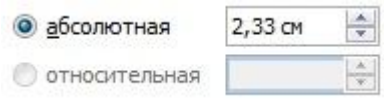


- *чекбокс* – вибір декількох варіантів одночасно:



- *комбобокс* – дані можна як впечатати самостійно, так і вибрати зі списку (рис. 6.6);
- *повзунок* – для уведення даних у певних границях, оптимальний для невеликої кількості варіантів:

- Поле редагування із *прокручуванням* – якщо варіантів багато:



Для більш ефективного діалогу необхідно завжди показувати границі діапазону в спливаючій підказці.

6.5. Кнопки

Кнопки розміщаються безпосередньо в інтерфейсі, не вимагаючи від користувача будь-яких зусиль, щоб їх побачити. Кнопки звичайно згруповані за змістом. Вони великі, читабельні, доступні й прості у використанні навіть для

непідготовлених користувачів. Однак вони займають багато місця, у порівнянні з меню.

Щоб інтерфейс був більш очевидним, кнопки групують за їхньою дією на певний зміст (контент).

Стандартні інструменти для WYSIWYG редакторів часто згруповані за функціями. На рис. 6.10 продемонстровані приклади з редакторів Word й Flash Builder, які показують декілька часто використовуваних інструментів у групах, що полегшує впізнавання.

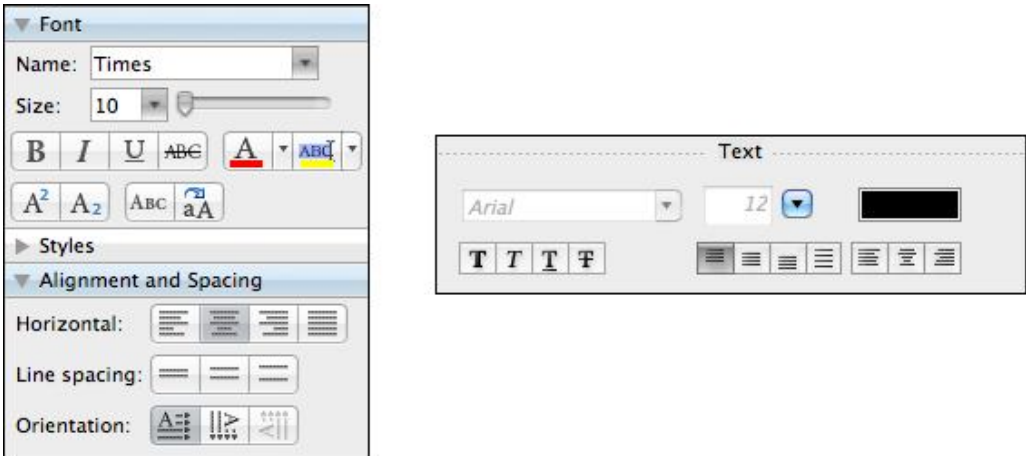


Рис. 6.10. Групування кнопок в Microsoft Word й Adobe Flash Builder

На рис. 6.11 показаний iTunes, в якому розміщено групи кнопок у кожному з 4-х кутів головного вікна, плюс стандартна смуга кнопок у заголовку для керування вікном, таких як закрити або зменшити.

У головному вікні iTunes не менше 13 кнопок, і це навіть без 4 кнопок при смугах прокручування або кнопок у заголовку. У цьому інтерфейсі багато можливостей, проте завдяки турботливій семантичній і візуальній організації цей інтерфейс не здається непереборним.

У веб-дизайні кнопки звичайно використовуються для відправлення даних, наприклад, після заповнення форми листа, реєстрації, пошуку й т. п. На стартових сторінках заклик до дії завжди виконується у вигляді великої, окремо розташованої кнопки, яка так і просить «Натисни мене!».

Рис.6.11. Групування кнопок на прикладі iTunes

Посилання – це кнопки, які не мають границі. Тому при проектуванні посилань особливо важливий зворотний зв'язок. Коли користувач проводить мишкою над текстом посилання, необхідно змінювати форму курсору й підкреслювати текст для посилення враження можливості натискання. Посилання завжди веде на іншу сторінку.

6.6. Функціональність клавіатури

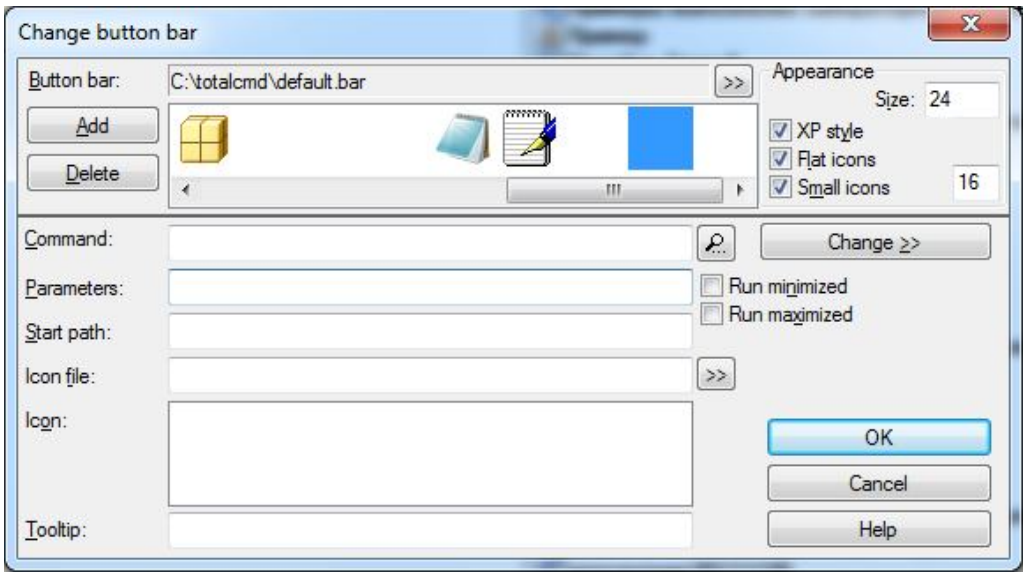


Рис. 6.12. Клавіші доступу

Клавіатурні скорочення, такі як Ctrl-S для збереження, варто проектувати в більшості десктопних додатків для доступності й ефективності використання. Більшість платформ із користувацьким інтерфейсом, включаючи Windows,

Mac, і деякі Linux, мають керівництва, які описують стандартні скорочення – й всі вони подібні.

До того ж, меню й елементи управління часто мають у назві підкреслену букву, які називають клавішами доступу. Натискання на однойменну клавішу клавіатури разом із клавішею Alt дозволяє звернутися до відповідних елементів інтерфейсу без миші й табулювання (рис. 6.12).

6.7. Правила взаємодії з об'єктом

Графічний пристрій введення (ГПВ) – це механізм для передачі системі інформації про певне місце розташування або вибору об'єкта на екрані монітору. Приклади ГПВ: миша, тачпад, трекбол і т. п.

Програма повинна бути розроблена таким чином, щоб від користувача не було потрібно майстерне володіння технікою позиціювання миші. Й на те є вагомі причини:

1. Іноді доводиться користуватися не самими оптимальними маніпуляторами – трекболами, трекпадами, які складніше контролювати, ніж звичайну мишу.

2. Іноді доводиться користуватися мишею не в самих сприятливих умовах: стіл завалений паперами, миша давно не чищена або сама миша такої якості, що курсор живе самостійним життям.

3. Іноді людина, яка сидить за комп'ютером, – новачок, моторні навички якого ще не достатньо розвинені для досконалого володіння мишею.

4. Деякі люди ніколи не зможуть розвити ці самі навички. Наприклад, страждаючі від артриту, тремору, кистьового тунельного синдрому, інваліди, маленькі діти або старі.

5. Багатьом просто не вдається двічі клацнути мишею, не пересунувши її при цьому на пару сантиметрів. У результаті вони часто тягають вікна по екрану, хоча насправді просто хотіли запустити додаток.

6. Деякі люди вважають, що постійне застосування миші сповільнює роботу.

Подвійний щиглик на елементі. Користувачі намагаються завжди застосовувати подвійного щиглика до елемента або, щоб відкрити його, або для чогось що залежно від контексту. У графічних редакторах, наприклад, подвійний щиглик приводить до відкриття вікна властивостей або спеціалізованого редактора. Подвійне натискання на іконку в більшості випадків запускає додаток. Подвійне натискання на текст дає можливість редагувати цей текст на місці.

Перетаскування. Тягти й кидати елементи інтерфейсу зазвичай означає або «перемістити це сюди» або «зробити це з тим». Інакше кажучи, хтось може перетягнути файл на іконку додатку, щоб відкрити файл за допомогою цього додатка. Або перетягнути файл із одного місця у файл-менеджері в інше, таким чином, копіюючи його. Перетаскування залежить від контексту, але це завжди виливається в один з подібних результатів.

Дії над контентом. *Зміст* (або *контент*) – це інформація, яка перебуває в комп'ютері або іншому пристрої, призначеному для обробки інформації, і є для користувача осмисленою й корисною.

Елементарні дії, виконувані користувачем у різних комбінаціях, породжують набір *елементарних операцій*, які застосовуються до змісту й використовуються майже у всіх інтерфейсах: указання, виділення, активізація, модифікація, генерація, видалення, переміщення, трансформація (перетворення в інший тип даних), копіювання, підсвічування. Зміст може бути відправлений або отриманий від зовнішнього пристрою, або скопійований в іншу область усередині системи. Наприклад, зміст можна роздрукувати, відправити електронною поштою, зберегти на жорсткому диску, копіювати в інший документ і т. п.

Кожна елементарна операція повинна завжди викликатися однаковим чином, незалежно від того, до яких об'єктів вона застосовується. В основному когнітивні розходження між програмами полягають у способах подання виділеного змісту й того, як користувач може оперувати ним.

Підсвічування (highlighting) означає, що за допомогою яких-небудь засобів відображеному на екрані об'єкту надається помітна відмінність. Підсвічування одиничного об'єкту при переміщенні курсору без якихось інших дій з боку користувача (як, наприклад, натискання на кнопку миші) є *указанням* (indication). За допомогою указання користувач може в будь-який момент знати, на які об'єкти він указує з погляду системи.

Підсвічування, використовуване для указання, не повинне бути занадто контрастним або яскравим, щоб рух курсору не викликав роздратування. Варто звернути увагу на те, що чим менше об'єкт, тим більший контраст повинен використовуватися для його указання.

Виділення (selecting) – це процес, за допомогою якого користувач указує, що один або кілька об'єктів мають особливий статус, який може бути сприйнятий системою. Як результат процесу виходить *вибірка* (selection). Звичайно користувач робить вибірку з метою застосування до неї в найближчому часі команди. На відміну від менш постійного указання, виділення, яке позначає вибірку, є більше стійким і зберігається, навіть якщо користувач відведе курсор убік.

6.8. Організація пошуку

Інтерфейси до засобів пошуку звичайно будуються на основі двох підходів. Найпоширенішим є *пошук з роздільниками* (delimited search), який зустрічається в більшості текстових процесорів. При типовому пошуку з роздільниками користувач включає режим, у якому будь-який уведений текст розглядається як шаблон для пошуку. Звичайно для цього використовується діалогове вікно, споряджене полями для уведення символів. Після виклику діалогового вікна користувач уводить комбінацію символів і роздільник, у якості якого звичайно використовується якийсь символ, заборонений до відображення в шаблоні. У більшості діалогових вікон користувач також може обмежити шаблон натисканням на кнопку Добре, Пошук, Знайти або Знайти наступне за допомогою ГПВ.

Коли об'єкт пошуку виявлений, він вибирається, а курсор розташовується відразу наприкінці вибірки.

Покроковий пошук (incremental search) зараз широко застосовується в інтернеті. Коли користувач уводить перший символ шаблону, система використовує цей символ як повний шаблон і відразу ж починає пошук першого екземпляру цього символу в обраному напрямку. Якщо екземпляр цього першого символу виявляється до того, як уведений наступний символ шаблону, то він позначається як вибраний, а курсор міститься відразу наприкінці вибірки.

Якщо ж наступний символ шаблону вводиться до того, як екземпляр буде знайдено, то цей символ додається до шаблону й пошук триває тепер уже відносно екземпляра розширеного шаблону. Процес повторюється в міру додавання символів до шаблона пошуку.

При використанні клавіші Backspace або Delete для видалення символів із шаблона покрокового пошуку, процес пошук вертається до попереднього екземпляру, знайденого за тим шаблоном, який був перед додаванням до нього наступного символу.

Пошук з роздільниками є менш зручним, тому що у випадку коли користувач зробив помилку в одній букві шаблону, йому доведеться чекати закінчення процесу пошуку, який завідомо закінчиться невдачею, щоб внести зміни.

6.9. Запрошення

Вдало розроблений дизайн хороший саме тим, що він дозволяє відразу зрозуміти, як об'єкт функціонує. Наведемо приклад із життя. На деяких дверях на рівні руки розташовані великі металеві пластини. Єдине, що можна проробити з ними – штовхнути. Як сказав Дональд Норман, вони *запрошують* вас штовхнути їх. На інших дверях ви побачите великі закруглені ручки, які просто змушують вас *потягнути* за них. Дизайн інших об'єктів продуманий не так добре й іноді буває важко припустити, для чого ж вони призначені. Самий

наочний приклад – коробки для зберігання CD, дизайн яких розроблений так, що людина повинна *особливим чином* розташувати пальці на її кришці й потягнути в *певному* напрямку. Але ніщо в дизайні не говорить про те, у якому ж напрямку тягти або яким конкретно чином потрібно покласти пальці.

Самий вдалий спосіб створити запрошення – викликати асоціацію з людською рукою в потрібному місці об'єкту.

Для цього на предметах створюються виїмки, накладки й т. ін., які позбавлені функціональності – їхнє призначення полягає в тому, щоб змусити користувача тримати предмет правильно.

Точно також і гарний комп'ютерний інтерфейс-дизайн застосовує різного роду запрошення. Різні відтінки сірих кольорів, використані в дизайні кнопок, надають їм опуклий вигляд. Не для того, щоб круто виглядати – опуклість кнопок *запрошує* до натискання. Сам вид кнопок припускає, що з ними потрібно робити, а саме – кликати по них. Зараз у моду знову ввійшов «плоский» дизайн. Кнопки виглядають круто, але змушують задуматися, на що можна натиснути. Особливо ця проблема актуальна у веб-дизайні.

Рис. 6.13 Дизайн кнопок, що приводить до плутанини

На рис. 6.13 кнопки GO і "LOG ON" виглядають опукло, таким чином зрозуміло, що по них можна кликнути. Кнопки SITE MAP і HELP позбавлені подібної доброзичливості, більше того, вони в точності нагадують дизайн псевдокнопки QUOTES, яка й взагалі-то кнопкою не є. Тут повністю порушений один з основних принципів інтерфейсу – погодженість.

Рис. 6.14 Запрошення «потягнути»

Стрілочка із крапок у нижньому правому куті вікна (рис. 6.14) запрошує потягнути за нього для зміни розміру вікна.

Один із кращих прикладів запрошення – діалог із закладками (tabbed dialog). Свою популярність він одержав у 1992 році.

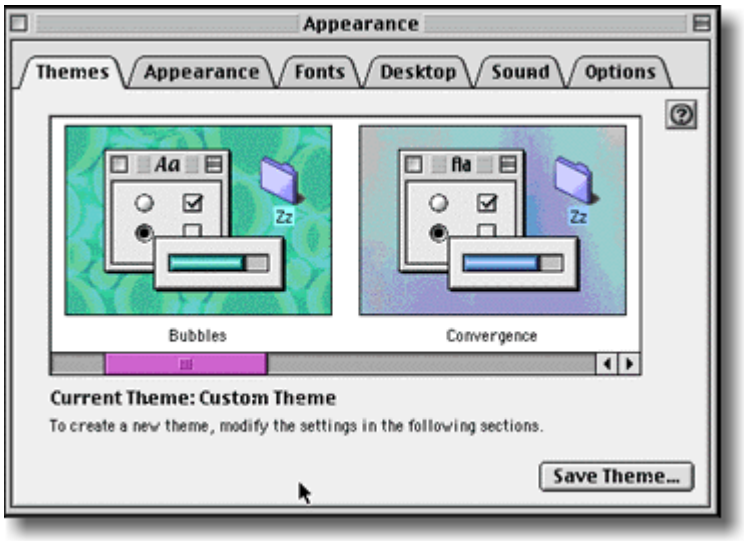


Рис. 6.15 Діалог із закладками

З рис. 6.15 очевидно, що є 6 закладок; очевидно, на який з них користувач зараз перебуває; очевидно, як перейти до іншої. При проведенні тестування, кожний учасник зрозумів, як користуватися інтерфейсом.

6.10. Основні функціональні елементи веб-інтерфейсів

Сьогодні практично всі сфери діяльності представлені в інтернеті. Компанії змушені конкурувати в тому числі й своїми сайтами. Будь-який дріб'язок незрозумілий користувачеві приведе до того, що він просто піде на інший сайт і скористається послугами іншої фірми. Тому технології створення простого й зрозумілого інтерфейсу веб-додатків найбільш актуальні.

Заходячи на сайт, людина повинна з одного погляду побачити основне призначення сайту.

Проектувальники, любовно розміщаючи елементи, часто розраховують на те, що користувач буде їх так само старанно розглядати й прочитає все, що написано на сторінці. Насправді користувачі переглядають сайти дуже швидко, як говориться «по діагоналі». Вони не читають, тому що завжди поспішають, уважають, що їм це не треба, і взагалі звикли так робити із журналами й газетами (переглядати тільки заголовки й картинки).

Найчастіше користувачі вибирають *не* оптимальний варіант, а той, котрий *першим здався придатним*. Як тільки людина натрапляє на посилання, яке, на її погляд, може привести до того, що вона шукає, то, найімовірніше, вона клацне по цьому посиланню. На пошук найкращого варіанта в користувача зазвичай немає часу, – якщо вибір виявився невдалим, його легко виправити. Ретельне зважування всіх варіантів не завжди приводить до позитивних результатів, і, взагалі, вгадувати цікавіше. Тому важливо створювати правильну ієрархію на сторінці, детальніше про це у темі 7.

Якщо в користувача будуть виникати при погляді на сторінку наступні питання, то навряд чи він ще повернеться на цей сайт:

- як тут усього багато, звідки мені почати?
- чому це так називається?
- чи можна сюди натиснути?
- де розділи сайту – те або це?
- навіщо тут це?
- це одне посилання або два?
- де я?
- що саме головне на цій сторінці?

Виключення можуть становити тільки сайти, які надають дуже рідкісні послуги або ексклюзивний контент.

Як можна було помітити більшість перелічених питань можна об'єднати в групу «Що де перебуває?», іншими словами в групу питань навігації по сайту.

Навігація. Люди не будуть користуватися сайтом, якщо їм не зрозуміло як по ньому переміщатися.

Переміщення починається з ухвалення рішення *що* шукати. Потім людина вибирає, чи буде вона шукати сама або звернеться до довідки. В 99% випадках вибір робиться на користь самостійного пошуку й тоді йде переміщення за вказівниками.

Основні навігаційні елементи (вказівники):

- назва сайту;

- назва сторінки;
- розділи;
- підрозділи;
- вказівник місцезнаходження;
- локальна навігація;
- сервіси;
- зменшена текстова версія всієї навігаційної системи.

Назва сайту повинна бути присутня на кожній сторінці. Традиційно вона розташовується у верхньому лівому куті. Назву сайту бажано виконати у вигляді логотипу – додати рисунок, написати особливим шрифтом, підібрати кольори – іншими словами зробити логотип впізнаваним. Якщо в користувача відкрито багато вкладок з різними сайтами, то логотип відразу допомагає визначити що це за сайт. На головній сторінці сайту логотипу може бути відведено значну роль, проте, якщо це не головна сторінка – немає необхідності робити його самим помітним елементом.

Логотип повинен відображати напрямок діяльності сайту. Краще як логотип використати тільки буквену назву, якщо неможливо підібрати картинку за змістом. Добре доповнити логотип слоганом, який кількома словами описує, що даний сайт або компанія робить. Особливо це важливо, якщо компанію поки ще не знають широко за назвою. Але навіть відомі компанії завжди планують залучати нових клієнтів, і тому такий слоган не перешкодить для відвідувачів, які прийшли на сайт уперше. Слоган повинен складатися з 5-7 слів, але бути інформативним. Варто відрізняти слоган від девізу, останній просто обрисовує світлі цілі й устремління.

Рис. 6.16 Приклади логотипів сайтів

На рис. 6.16 показані логотипи трьох різних сайтів. Перший логотип має основу у вигляді назви сайту E-PEPPER, підпис й малюнок, який ніяк не характеризує електронну комерцію і більше підійшов би для сайтів пов'язаних з дизайном. Хоча підпис і не є слоганом, але він дуже чітко пояснює призначення сайту. Другий логотип належить англомовному сайту Цей сайт є форумом типу питання-відповідь для програмістів, які зіштовхуються із проблемами у своїй роботі. З одного боку логотип містить і буквену назву сайту stackoverflow (що переводиться як стек переповнений) і навіть зображення, яке й демонструє переповнений стек. Треба помітити, що переповнений стек є проблемою в програмуванні. Але зіставити це все разом можна тільки прочитавши в у протилежному куті сайту наступне повідомлення: «This is a collaboratively edited question and answer site for**professional and enthusiast programmers**. It's 100% free, no registration required.» (Це спільно створюваний сайт питань і відповідей для професійний програмістів і ентузіастів. 100% безкоштовний, не вимагає реєстрації (англ.)). Якби слова питання, відповідь і програмування якось пролунали в логотипі, було б набагато зрозуміліше призначення сайту.

Останній приклад – логотип відомої пошукової системи. Тут нічого зайвого й усе на своєму місці – і буквений логотип з впізнаваною буквою Я на початку, і коротенький слоган, який відображає призначення сайту.

Назви веб-сторінок можна порівняти з дорожніми знаками.

Вимоги до назв:

- кожна сторінка повинна мати назву;
- назва сторінки повинна обрамляти розташований на ній зміст;
- назва сторінки повинна відповідати назві посилання, за яким здійснений перехід. Назва посилання може бути небагато коротше за назву сторінки, але зміст не повинен губитися;
- назва веб-сторінки повинна відповідати назві у вкладці браузера.

Існує поняття постійної або *глобальної навігації* (рис. 6.17), яка з'являється на кожній сторінці сайту. Це дає користувачеві впевненість, що він не загубиться. Виключення становлять сторінки заповнення форм, уведення

реєстраційних даних. Для таких сторінок треба мати скорочену версію глобальної навігації. Звичайно з таких сторінок іде перенаправлення на попередню сторінку. *Локальна навігація* – це список розділів одного рівня.

Рис. 6.17 Елементи глобальної навігації на прикладі сайту zverenki.com

Основні елементи глобальної навігації:

- назва сайту;
- панель розділів;
- сервіси;
- посилання на початкову сторінку;
- посилання на сторінку пошуку.

Панель розділів – це основна навігація. Може містити *підрозділи* – вторинну навігацію.

Сервіси – посилання на важливі компоненти сайту, не належні ієрархії. Це може бути допомога, карта сайту, зв'язок з адміністрацією сайту, перехід на головну сторінку, вхід, реєстрація, особистий розділ, новини, питання, які задають часто, і т. п. Як правило, навігація містить чотири-п’ять сервісів.

Навігація головної сторінки може відрізнятися від навігації на інших сторінках: містити опис розділів, мати іншу орієнтацію, мати більше місця для розпізнавальних знаків. Важливі елементи головної сторінки не вимагають якихось особливих ілюстрацій, вишуканих рамочок і кричущих кольорів. Відвідувачі часто не звертають увагу на графіку, вважаючи її рекламними банерами, і зосереджують свою увага на тих частинах головної сторінки, які, як їм здається, мають відношення до справи.

Якщо користувач хоче повернутися до чого-небудь, то йому потрібно згадати, де це перебуває в рамках структурної ієрархії й концепції сайту, й потім повернутися в те місце. Саме із цієї причини закладки, або персональні інтернет-ярлики, які зберігаються користувачем, мають таке велике значення, і саме тому по кнопці «Назад» робиться від 30 до 40 відсотків всіх натискань в Інтернеті.

Це також пояснює, чому посилання на початкову сторінку сайту настільки важливе. Початкова сторінка – це порівняно фіксована позиція. Для користувача початкова сторінка на сайті виконує ту ж роль, що й Полярна зірка для мандрівника. Якщо щось не виходить, завжди можна повернутися у вихідну точку сайту – початкову сторінку.

Щоб у користувача не було відчуття втрати простору потрібно оформляти навігацію за типом карти в парку «Ви знаходитесь тут». Це можливо зробити виділенням у навігаційних лінійках *поточного місцезнаходження* (рис. 6.18).

Рис. 6.18 Вказівник поточного місцезнаходження на сайті Bonprix

Ще один тип покажчика – «*хлібні крихти*» (рис. 6.19). На відміну від вказівників «Ви знаходитесь тут», за якими користувач може визначити свою позицію усередині ієрархії сайту, «хлібні крихти» показують тільки шлях від початкової сторінки до поточного місцезнаходження.

Рис. 6.19 «Хлібні крихти» на сайті Bonprix

Інакше кажучи, одне показує, де користувач перебуває в загальній схемі сайту, інше – як туди добратися. Для більшості сайтів одних тільки «хлібних крихт» недостатньо, щоб створити повноцінну навігаційну систему. З їхньою допомогою неможливо повністю показати ієрархію сайту в її верхніх (хоча б двох) рівнях. Вони можуть указати тільки деякий напрямок.

Рекомендації з використання «хлібних крихт»:

- «хлібні крихти» розміщаються у верхній частині сторінки;
- між назвами рівнів для розділення використовується знак «>»;
- застосовується невеликий розмір шрифту;
- як пояснення використовуються слова «Ви перебуваєте тут»;
- для останнього елемента в списку застосовується жирне накреслення;
- недобре використовувати «хлібні крихти» замість назви сторінки.

Зменшена версія веб-навігації зазвичай розташовується внизу сторінки сайту під всім контентом (цю частину сторінки називаються *футером*) й виконується дрібним шрифтом. Наявність даного елементу на сторінці є ознакою хорошого тону, проте не є обов’язковим.

Тема 7. **Візуальна організація. Використання звуків й анімації.**

Визначення «мови візуального подання» – це підбір кольорів, стилів, шрифтів та інших «матеріальних» властивостей інтерфейсу.

7.1. Основи розмітки сторінки

Візуальна ієрархія: Що важливо? Що зв'язано? Як зробити, щоб речі виглядали важливими: змінити розмір тексту? Виділити текст кольорами? Зробити контрастними кольори фону під текстом? Використання двох подібних ознак відокремлює текст від основного.

Для того щоб зробити або усунути акцент на більше дрібних елементах, таких як основний текст посилання, використаються наступні прийоми:

1. *Щільність* – блоки з більш жирним шрифтом, або меншим міжрядковим і міжбуквеним інтервалом виглядають контрастно.

2. *Кольори фону* – чим більше контраст, тим більше залучається увага. Найдужчий контраст у чорного й білого.

3. *Положення й розмір* – великі або середні текстові блоки в центрі сторінки привертають увагу як основний зміст (контент). А маленька смужка тексту внизу сторінки як би говорить – я просто футер і мене можна проігнорувати (рис. 7.1).

Рис. 7.1 Розмір тексту в ієрархії

4. *Ритм* – списки, таблиці, рядки заголовків і виводів, відділення пробілами створюють сильний візуальний ритм (рис. 7.2).

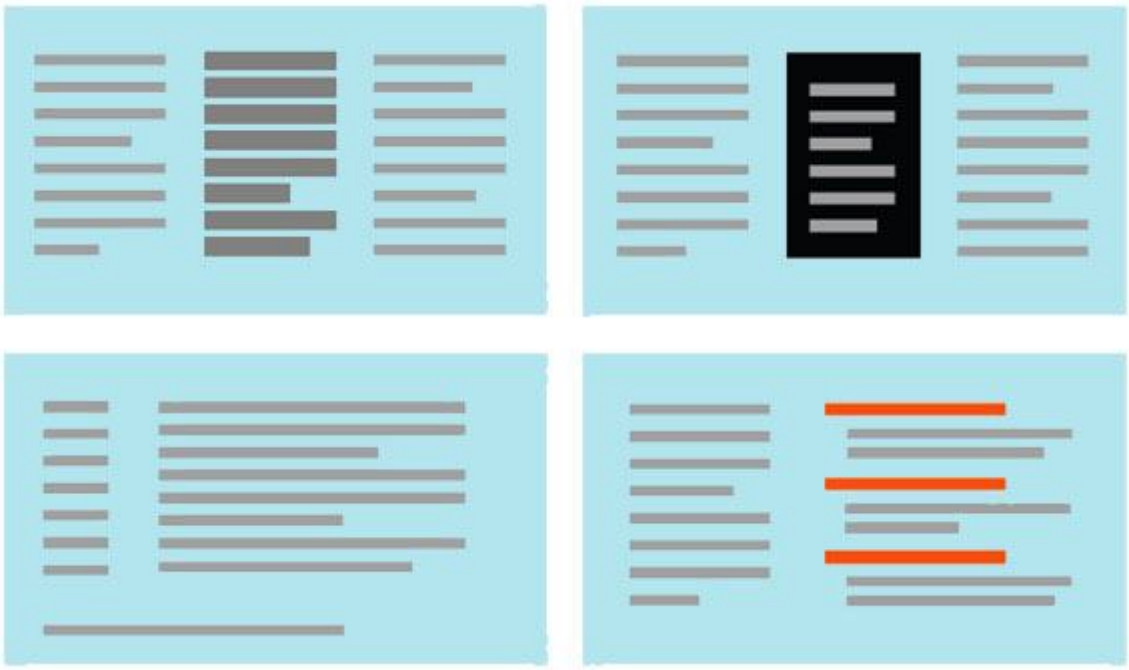


Рис. 7.2 Способи акцентування уваги на текстових блоках

Маленькі, але важливі елементи розміщаються вгорі сторінки, уздовж лівого краю або у верхньому правому куті. Вони повинні бути контрастними й візуально значимими, відділеними відступами від іншого.

На завантажених текстом сторінках, притаманних більшості сайтів, елементи керування – особливо поля пошуку, поля авторизації й великі кнопки повинні перебувати відособлено. Якщо людина шукає блок пошуку, то вона автоматично вибирає перше текстове поле, не читаючи мітки біля нього (рис. 7.3).

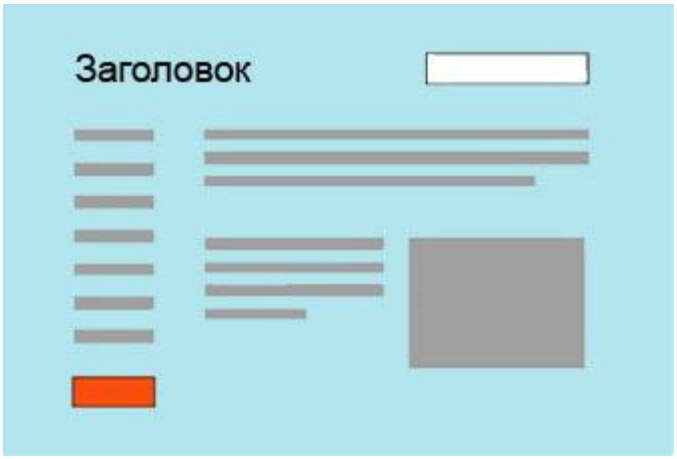


Рис. 7.3 Розміщення інформації на завантажених текстом веб-сторінках

Як показати відносини між елементами сторінки. Угруповання елементів використовується, щоб показати, що вони зв'язані, й навпаки, відстань показує відокремлені елементи.

Якщо потрібно показати рівність елементів для інтересів користувача використовуються блоки однакової висоти й стилю. Щоб показати, що один має значну перевагу або обраний, використовується незначна зміна кольорів фону або елемент обводиться лінією.

Список подібних елементів, упорядкований у рядок або колонку, подається набором рівних елементів, показаних у певному порядку. Такі елементи (маркіровані списки, навігаційні меню, текстові поля у формах, рядки таблиць, списки заголовків) повинні створювати візуальну лінію (рис. 7.4).

Стислий текст із відступом під сильним елементом показує, який елемент основний. Опис картинки, другорядний текст, коментарі використовують цей метод (рис. 7.5).

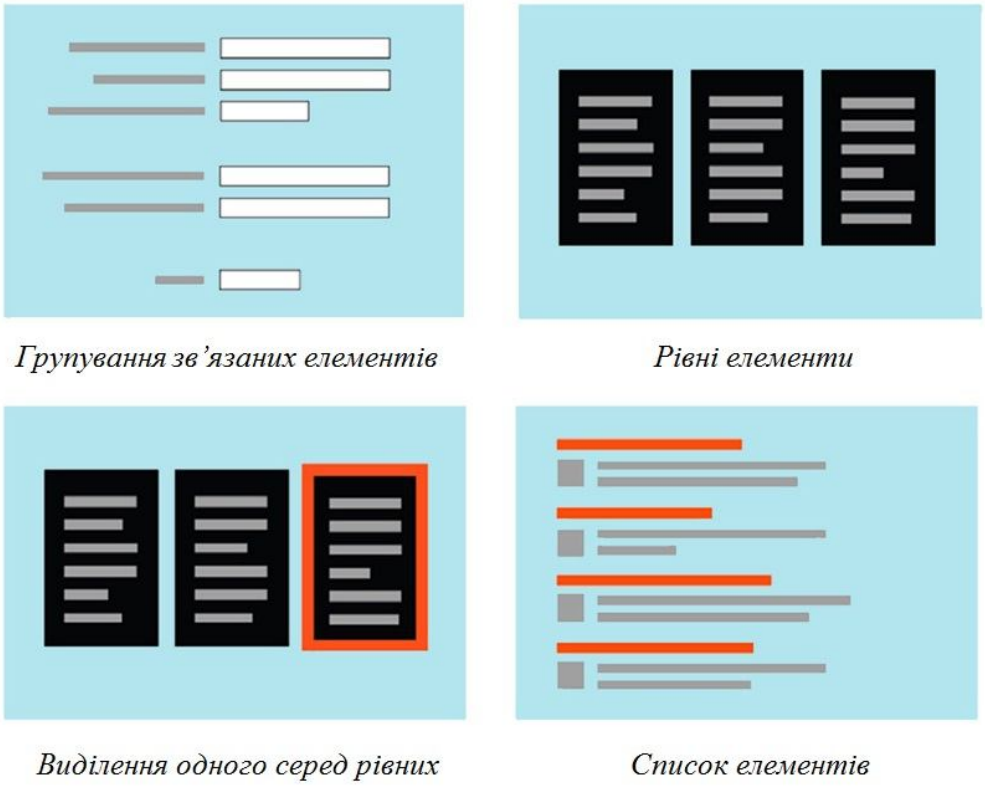


Рис. 7.4 Вирівнювання елементів сторінки за уявною лінією

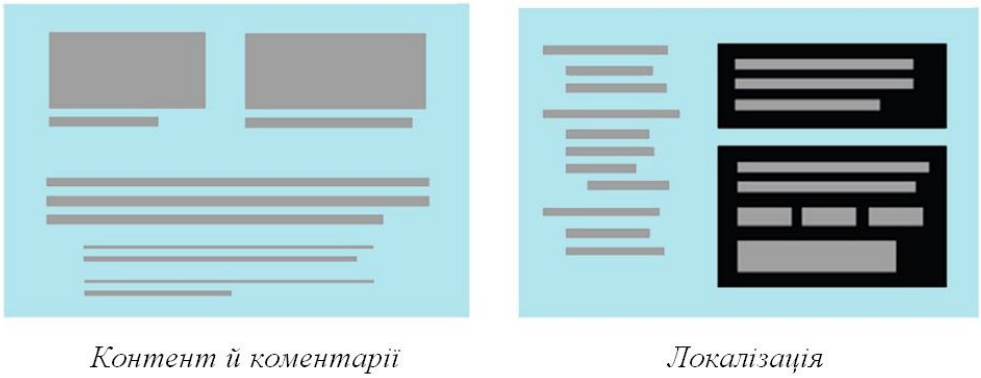


Рис. 7.5 Використання відступів

Локалізація має на увазі відносини батько/дитина. Для основних і сусідських відносин елементів, застосовуються контейнери, блоки кольорів заднього фону, модульні панелі, або угруповання відділенням порожнім простором. Відступи так само використовуються для ієрархічного меню.

4 принципи цілісності (Гештальта). Теорія про угруповання й вирівнювання була розроблена в XX столітті фізіологами-гештальтами. Вони описали кілька особливостей розташування, які вшиті у візуальну систему людини. Серед них:

1. *Сусідство – при розташуванні* елементів близько один одного, читачі проводять асоціацію між ними. Це основа угруповання контенту й елементів управління в графічних інтерфейсах.

2. *Подібність* – якщо дві речі однієї форми, розміру, кольору або орієнтації, читачі так само будуть асоціювати їх одну з одною.

3. *Поздовженність* – очі людини бажають бачити протяжні прямі й криві лінії, сформовані розташуванням маленьких елементів.

4. *Замкненість* – людина також хоче бачити прості замкнені форми, такі як квадрати, плями білого простору. Угруповання речей часто застосовуються, щоб створити замкнені форми.

Ці принципи добре використовувати як індивідуально, так і в комбінації один з одним.

Угруповання елементів за рахунок вільного простору між ними є переважніним перед поділом лініями або прямокутниками, тому що знижує візуальний шум на сторінці.

7.2. Візуальний плин: Що мені варто подивитися далі?

Візуальний плин вивчає шлях, за яким рухаються очі користувача під час сканування сторінки. Це тісно пов'язане з візуальною ієрархією, тому що добре спроектована ієрархія встановлює фокусні точки в тих місцях, де потрібно приділити увагу особливо важливим елементам. І візуальний плин веде погляд від них до найменш важливих. Кілька обставин працюють один проти одного при створенні візуального плину. По-перше це тенденція більшості людей читати зліва направо й зверху вниз. Якщо два елементи розташовані поруч (наприклад, текст і фото), передбачається наявність зв'язку між ними. Цей зв'язок сильніше, якщо один елемент розташований праворуч від іншого. На рис. 7.6 зображення відноситься до тексту під ним, це може спантеличити.

Якщо на сторінці є сильні фокусні точки, вони порушують звичний процес, як у кращий, так і в гірший бік.

Фокусні точки це місця, які приковують погляд користувача опріч його бажання. Він проходить по них від найбільш сильних до найбільш слабких. На грамотно спроектованій сторінці їх мало – занадто багато фокусних точок зводять нанівець важливість одна одної. Хороша візуальна ієрархія використовує фокусні точки, щоб провести погляд по потрібних точках у потрібному порядку.

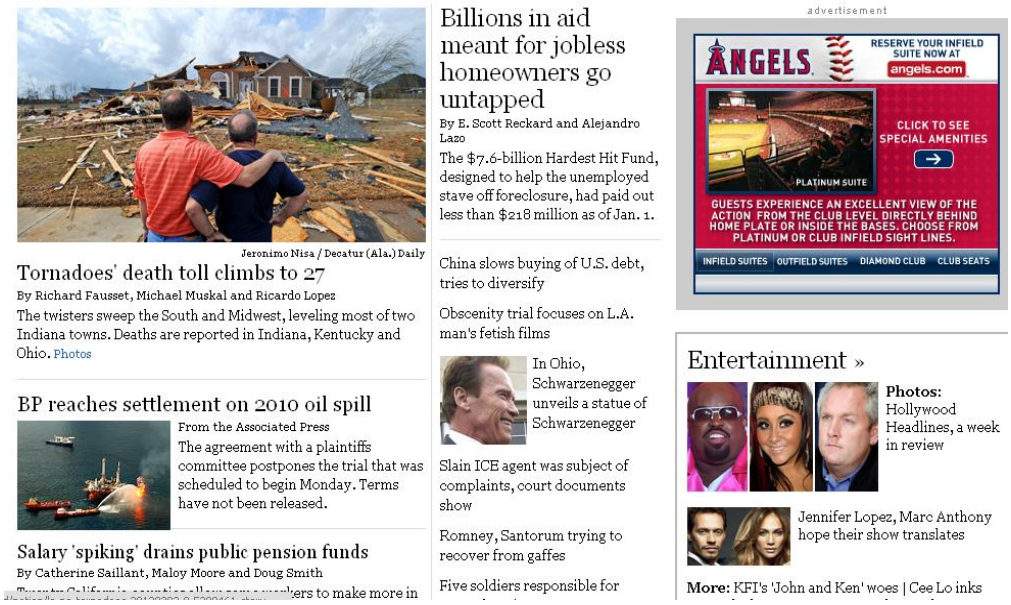


Рис. 7.6 Порушення візуального плину

Одним зі шляхів створення візуального плину є створення умовних ліній, прямих або хвилястих для зв'язування елементів на сторінці. Це створює візуальний напрямок проходження (рис. 7.7).

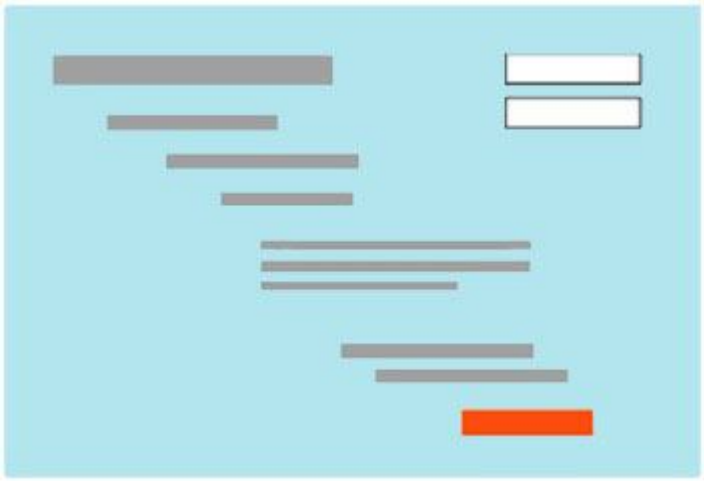


Рис. 7.7 Умовні лінії

Елемент управління (наприклад, кнопка) ставиться після тексту, якщо потрібно, щоб користувачі спочатку прочитали. Якщо це не важливо, то елемент керування ізолюється (рис. 7.8).

Засоби контролю повинні розташовуватися послідовно уздовж однієї лінії, кнопка типу «Я закінчив» (Так, Відмінити, Відправити ,Підтвердити, Купити й т. п.) завжди ставиться наприкінці.

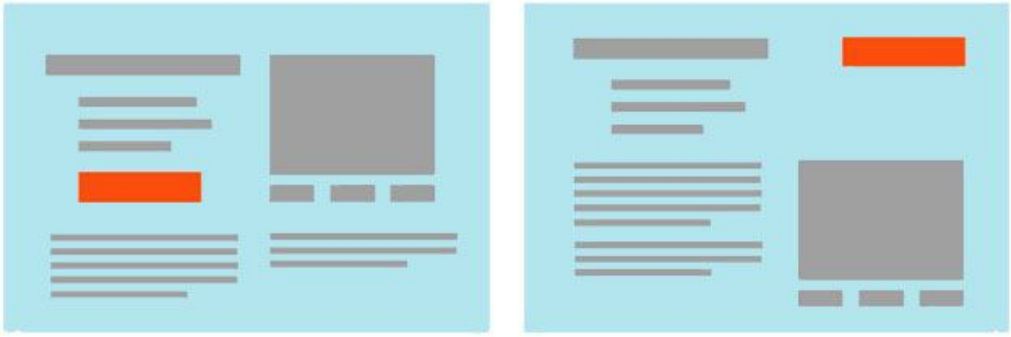


Рис.7.8 Заклики до дії за плином й проти нього

Для форм рекомендується використовувати вирівнювання по правому/лівому краю (рис. 7.9).

Рис. 7.9 Вирівнювання у формах

При використанні асиметричного розташування, важливо витримати баланс по горизонталі. Верхній лівий і нижній правий кут повинні бути врівноважені (рис. 7.10).

Нескладно зробити розмітку, яка добре діє на плин, але треба стежити й за тим, щоб не встановити таку, котра йде врозріз із плином. Наприклад, не

розкидати засоби контролю по всій формі, – від користувача буде потрібно багато зусиль, щоб знайти їх.

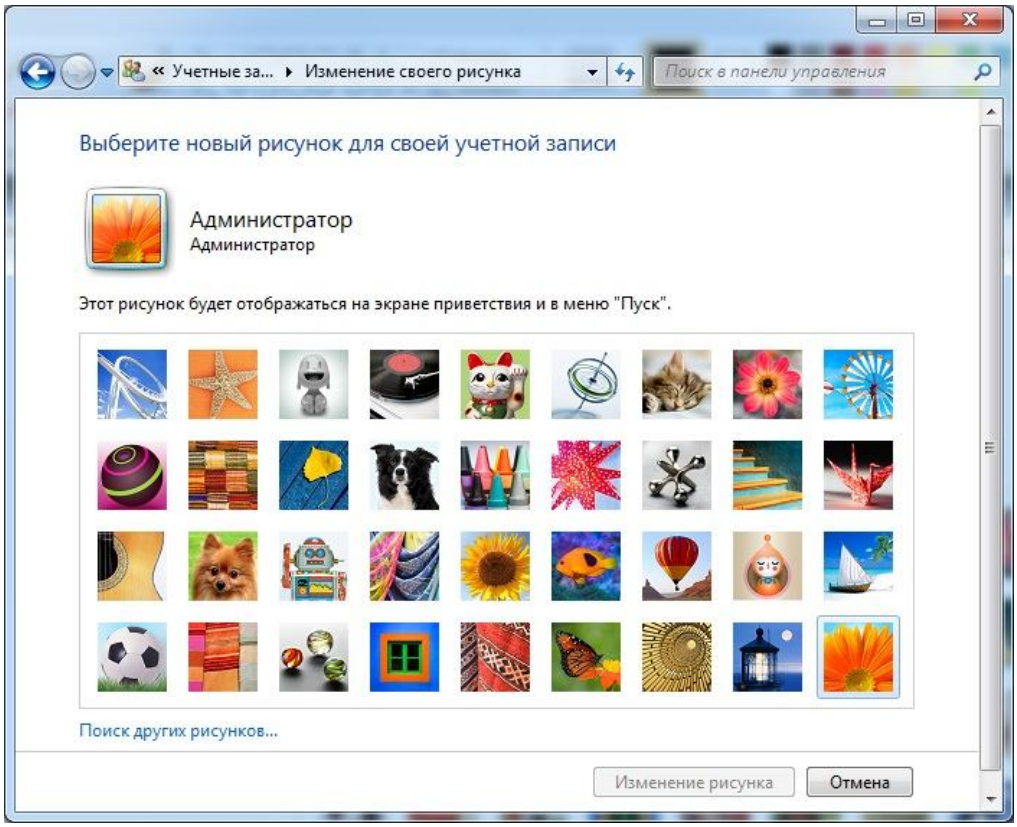


Рис. 7.10 Асиметрія в розташуванні елементів

Рис.7.11 Безладдя у візуальній ієрархії

На рис. 7.11 показаний приклад поганої ієрархії й поганого плину. Скільки тут фокусних точок? Як вони співвідносяться одна з іншою? Де очі користувача повинні зупинитися в першу чергу? Чому?

7.3. Підбір екранних шрифтів

Шрифт повинен бути досить великим, щоб його можна було читати без напруги.

Деякі шрифти здаються крупніше в порівнянні з іншими. Це пов'язане з x-висотою (рис. 7.12, 7.13).

Рис. 7.12 Параметри шрифтів

Всі шрифти на цьому рисунку мають однаковий розмір, проте деякі здаються більшими, тому що x-висота у різних сімействах шрифтів відрізняється.

Всі шрифти на цьому рисунку мають однаковий розмір, проте деякі здаються більшими, тому що x-висота у різних сімействах шрифтів відрізняється.

Всі шрифти на цьому рисунку мають однаковий розмір, проте деякі здаються більшими, тому що x-висота у різних сімействах шрифтів відрізняється.

Всі шрифти на цьому рисунку мають однаковий розмір, проте деякі здаються більшими, тому що x-висота у різних сімействах шрифтів відрізняється.

Рис. 7.13 Приклади шрифтів з різною x-висотою

Звичайних дизайнерів учать тому, що *пропорційні* шрифти більше елегантні, витончені, і, до того ж, легше читаються. Тільки все це стосується паперу, а не екрану. При редагуванні тексту перевагу одержують *моноширинні*

шрифти: такі вузькі символи як «l» й «i» в них більш помітні, й їх легше розрізнити.

На рис. 7.14 у першому випадку використовується шрифт Arial. Ширина букв «l» й «i» становить буквально один піксел. Різниця між маленькою «i» і «1» теж в один піксел. (Аналогічно уловити різницю між прописними «rn» і «m» майже неможливо, тому в поле редагування могло бути написане й *Fillrnore*).

Проблеми виникнуть і з миготливим курсором, у випадку коли потрібно вибрати одну букву у слові, тому що ширина курсору лише два піксели. У другому випадку використаний шрифт **Courier Bold**, що дозволяє уникнути всіх перерахованих вище проблеми.

Рис. 7.14 Пропорційний і моноширинний шрифти у полі редагування

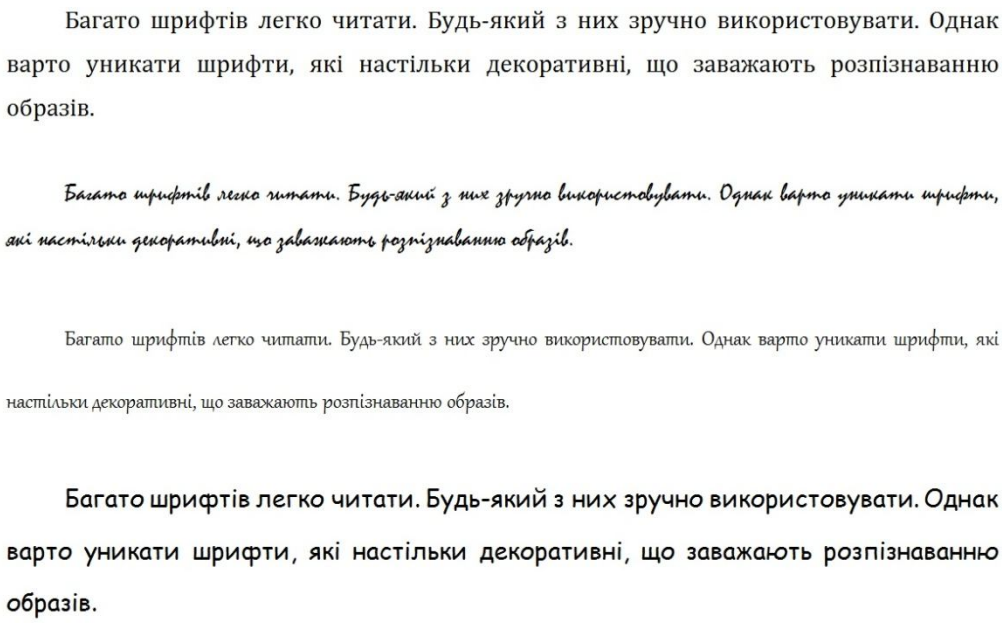


Рис. 7.15 Шрифти з різним ступенем декоративності

Не втухають суперечки про те, який шрифт краще й легше для сприйняття або більше відповідає духу часу. В одній з таких суперечок обговорюються два типи шрифтів – із зарубками й без зарубок. Прихильники шрифтів без зарубок, говорять, що такі шрифти більш прості і ясні. Супротивники заперечують, що

зарубки направляють око читаючого до наступної букви, тому сприймаються легше.

Дослідження показали, що різниці між ними в сприйнятті немає. Якщо тільки шрифт не містить надлишкову кількість завитушок. Люди не зберігають в пам'яті всі варіанти написання букв. Замість цього в мозку формується образ букви. Коли людина бачимо щось подібне, її мозок впізнає цей образ.

7.4. Колірне коло: застосування. Колірні гармонії

Для підбору колірних сполучень часто використовується колірне коло Іттена (рис. 7.16).

Існує кілька класичних комбінацій кольорів.

Комплементарними, або додатковими, контрастними, є кольори, розташовані на протилежних сторонах колірного кола Іттена. Краще такі комбінації виглядають, якщо кольори соковиті. Наприклад, червоний і зелений. Використовуються, коли потрібно щось виділити, підкреслити. Украй не рекомендується *використовувати комплементарні кольори для текстових композицій*.

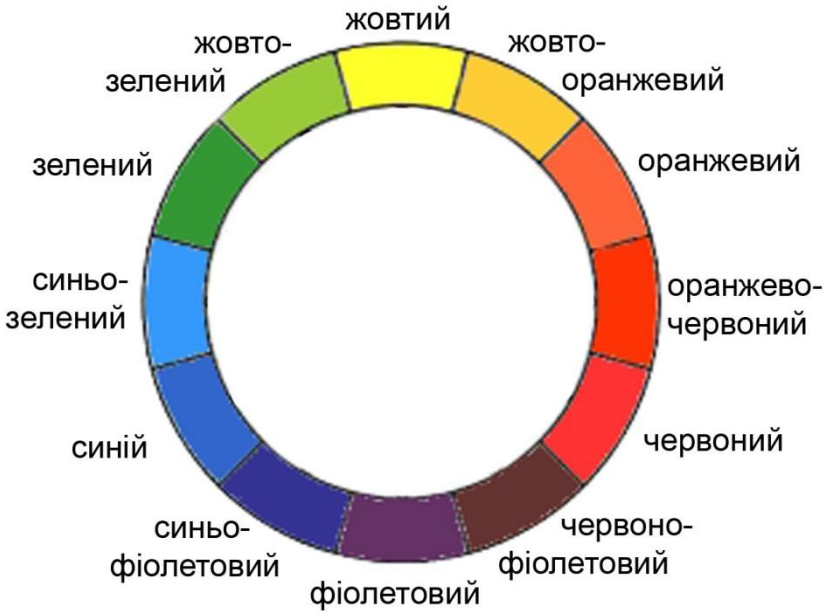


Рис. 7.16 Колірне коло Іттена

Класичну тріаду утворюють три рівновіддалених у колірному колі Іттена кольори, наприклад зелений, фіолетовий і оранжевий. Така композиція

виглядає досить живою навіть при використанні блідих і ненасичених кольорів. Для гармонійності в тріаді один колір використовується як основний, а два інших для акцентів.

Аналогову тріаду становлять три сусідніх кольори кола Іттена. При використанні цієї схеми, один колір вибирається головним, другий – підтримуючим, а третій використовується для акцентування. Також варто подбати про достатній контраст в аналоговій композиції. Добре використовувати аналогову схему для різних станів елемента інтерфейсу, наприклад, зелена кнопка при наведенні на неї курсором миші стає темно-зеленою, а при натисканні на ній – блакитною.

Контрастна тріада – варіант комплементарного сполучення кольорів, тільки замість одного протилежного кольору використовуються два сусідні до нього. Тобто на противагу зеленому ставляться червоно-фіолетовий і червоно-оранжевий. Виглядає ця схема майже настільки ж контрастно як зелений-червоний, але не настільки напружено.

Прямокутна схема складається із чотирьох кольорів, кожні два з яких – комплементарні (наприклад, червоний і жовтий проти зеленого й синього). Ця схема дає, найбільшу кількість варіацій вхідних у неї кольорів. Щоб простіше було збалансувати прямокутну схему, один колір вибирається домінуючим, інші – допоміжними.

Квадратна схема практично повторює прямокутну схему, але кольори в ній рівновіддалені по колу (червоний, синій, зелений жовтий). Тут також варто вибрати один домінуючий колір.

Варто враховувати при використанні чотирикутних схем, що, складаючи їх, кольори не можна розташовувати поруч. Наприклад, сині букви на червоному фоні. Так само як і подібне сполучення комплементарних кольорів – це дуже сильний удар по очах.

Взагалі, щодо текстів, то найкращим поєднанням вважається чорні букви на білому фоні, найгіршим навпаки – білі букви на чорному фоні. Якщо дуже потрібно застосувати останню комбінацію, то її можна пом’ягшити за рахунок

заміни білих букв на світло-світло-сірі. Стосовно інших кольорів слід запам’ятати, що розташування тексту поверх неоднорідного фону є дурним тоном. З точки зору зручності читання текст повинен розміщатися на однотонному фоні.

7.5. Зниження шуму на сторінці

Часто виникає ситуація, коли для рівнозначних елементів контенту необхідно виконувати невелику кількість однакових дій, наприклад видалення кожного елемента зі списку окремо. Щоб видалення було можливо в одну дію, має сенс розмісити біля кожного елементу списку свій функціонал видалення. Але це приведе до значного шуму на сторінці при великій кількості елементів, коли не буде зрозуміло, до якого елементу яка кнопка ставиться. Вирішується дана проблема за допомогою *інструменту наведення* (Hover Tools). Доступні дії стають видимі тільки для конкретного елементу списку після наведення на ньому вказівника (рис. 7.17).

Перевага даного інструменту – ясність інтерфейсу, але побічним ефектом є те, що користувач не бачить відразу доступні функції. Компромісний варіант використаний у додатку на рис. 7.18, коли доступні дії видимі відразу, але елементи управління здаються неактивними доти на одному з них не наведено курсором.

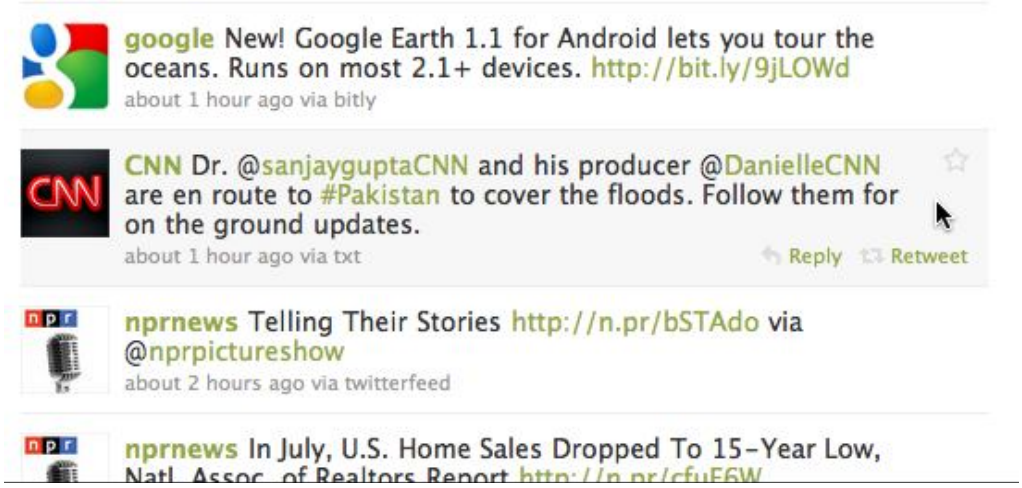


Рис. 7.17 Інструмент наведення (Hover Tools)

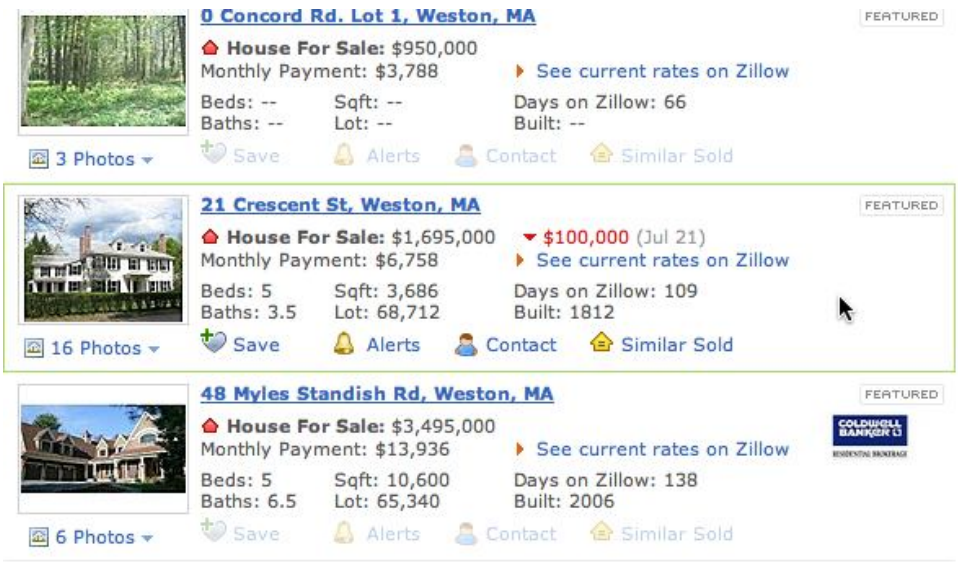


Рис. 7.18 Альтернативний варіант використання інструмента наведення

7.6. Звуки й інтерфейс.

Очікування, пов'язані із частотою, впливають на увагу. Люди підсвідомо будують ментальні моделі щодо того, як часто відбувається та або інша подія. Якщо розробляється продукт або додаток, у якому користувачі повинні помітити появу виняткової події, рекомендується використати сигнал (звуковий або візуальний) для залучення уваги.

Слухову форму пред'явлення інформації варто використати в наступних випадках:

- для сигналів небезпеки, тому що слух, на відміну від зору, не здатний до мимовільного самовимикання;
- при перевантаженні зору;
- коли робота оператора вимагає його постійного переміщення, й інформація повинна прийматися незалежно від орієнтації голови оператора;
- при обмеженні зору зовнішніми або внутрішніми умовами;
- у специфічних умовах (аноксія, стан невагомості, вплив перевантажень і т. п.);
- коли в повідомленнях системи мова йде про події, які розвиваються у часі;

- при необхідності виділення сигналу із шуму, тому що слуховий аналізатор хороший детектор періодичних сигналів на фоні шуму.

Розрізняють звукові й шумові сигнали, з одного боку, й мовні – з іншої.

Використання звукових і шумових сигналів рекомендується в наступних випадках:

- при отриманні простого й короткого повідомлення, не пов'язаного з наступними повідомленнями;
- коли повідомлення вимагає негайної дії;
- коли оператор спеціально навчений розумінню змісту закодованого повідомлення;
- якщо оператор перевантажений мовними сигналами;
- якщо необхідно дотримання таємниці;
- коли оператор працює в групі;
- при сильних акустичних перешкодах.

Звук вибирається відповідно до тієї «кількості» уваги, яку потрібно одержати. Сигнали, які сильно привертають увагу, варто зарезервувати для дійсно важливих випадків, наприклад, коли користувач збирається відформатувати свій жорсткий диск або зробити дію, яку не може бути скасовано.

7.7. Анімація в інтерфейсі

Операції, які вимагають часу на виконання, переривають користувацький інтерфейс або запускаються на задньому плані на 2 секунди або більше потребують маскування анімацією (рис. 7.19).

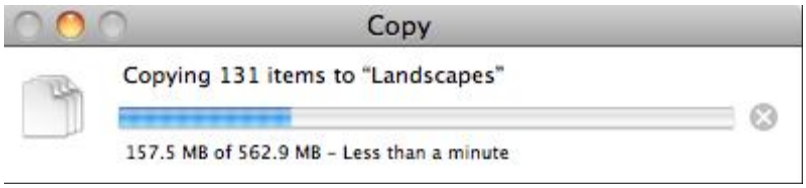


Рис. 7.19 Використання анімації для операції копіювання

Важливо показати користувачеві, скільки часу йому доведеться чекати. Згідно результатів експериментів, коли користувачі бачать, як дещо

відбувається, вони більш терплячі, навіть у випадку, коли їм треба чекати довше, ніж вони чекали без індикатора прогресу. Стрічка прогресу (рис. 7.19) є кращою за будь-який значок, що ідентифікує про затримку, наприклад курсор з годинником. Проте навіть просто значок краще ніж нічого. Тому що користувачі повинні знати, коли система «думає». Важливо забезпечити спосіб негайно скасувати операцію, яка вимагає часу, без наслідків.

Анімація дозволяє керувати увагою користувачів, направляти їх погляд на ті або інші області екрану, встановлювати візуальні й ідейні зв'язки, які дозволяють розуміти й запам'ятовувати те, що відбувається усередині користувацького інтерфейсу. Наприклад, якщо в результаті операції повинні бути з'єднані компоненти графічного вікна, краще показати, як це робиться. Наприклад, за допомогою лінії, яка рисується автоматично.

Анімація може й дратувати. Скріпка з одного відомого текстового редактору, яка весь час посміхається, яка постійно гойдається, обертається й усіляко залучає до себе увагу, перші дві хвилини здається забавною, а потім починає дратувати. За рахунок здатності завжди залучати до себе увагу, будь-яка анімація сповільнює роботу користувача. Тому здатність анімації привертати увагу варто використати тільки у випадках гострої необхідності.

Питання, на які може відповідати анімація:

- що це таке?
- звідки я прийшов і куди я йду?
- де я перебуваю?
- що я можу тепер зробити?
- як я це роблю?
- що відбувається?
- що я зробив?
- чому це відбулося?

При вирішенні людиною нудного завдання, необхідно підвищити рівень збудження користувача за допомогою звуку, кольору або руху.

При вирішенні складного завдання, необхідно знизити збудження, прибравши будь-які відволікаючі елементи, такі як кольори, звук, або рух, крім тих, що безпосередньо пов'язані з виконуваним завданням.

7.8 Особливості проектування для мобільних пристроїв.

Проблеми мобільного дизайну. При проектуванні інтерфейсів для мобільних пристроїв, особливо телефонів, потрібно враховувати, що користувачі не сидять перед великим екраном і не мають повноцінної клавіатури.

Тому важливо враховувати наступні особливості мобільних пристроїв:

1. *Маленький розмір екрана.* Мобільні пристрої звичайно не надають багато простору для демонстрації інформації або вибору функцій. На жаль, відсутня можливість розміщення бічних панелей, довгих головних меню, великих декоративних зображень або довгих списків посилань. Тут доводиться залишати в проекті інтерфейсу тільки саме основне, забираючи чисто дизайнерські елементи. При проектуванні сайтів для мобільних пристроїв варто залишати на головній сторінці тільки найбільш важливі функції, а від інших або позбутися зовсім, або сховати глибоку у ієрархії додатку.

2. *Різноманітність розмірів екрану за шириною.* Дуже важко проектувати інтерфейс, який добре працює для трьох різних екранів із шириною 128, 320 та 600 пікселей, тим більше, що цією шкалою не обмежується, є ще й проміжні варіанти. Прокручувати екран по вертикалі не занадто обтяжливо для користувачів. Тому використовувати екран у висоту можна вільно. А от використовувати доступний простір у ширину потрібно дуже акуратно. Іноді навіть має сенс створити різні версії для кожного типу пристроїв – для планшетів, для смартфонів і т. п.

Ширина екрану визначається не тільки сантиметрами, але, головним чином, щільністю пікселей і властиво розмірами окремо взятого піксела. При проектуванні важливо пам'ятати, що не всі пікселі однакового розміру (рис. 7.20).

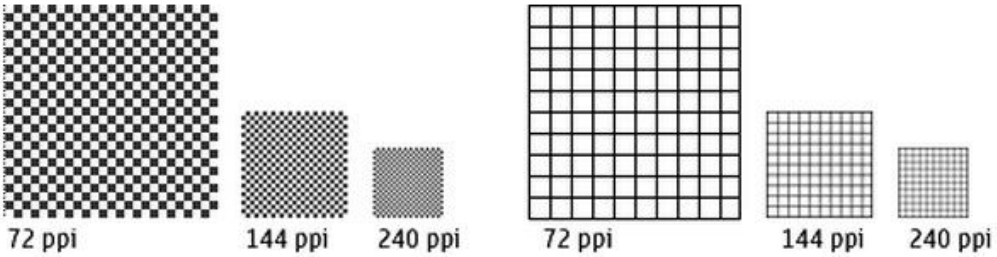


Рис. 7.20 Залежність ширини зображення від розміру й щільності пікселей (ppi – від англ. pixel per inch – пікселей на дюйм)

Необхідними діями при проектування інтерфейсів для мобільних пристроїв є:

- визначення повної шкали щільності пікселей, яку повинен підтримувати додаток;
- перегляд розроблених варіантів інтерфейсів на пристроях із різною щільністю пікселей з метою переконання, що найбільш важливі деталі не губляться;
- визначення й проектування базового макету, заснованого на відносних величинах (наприклад, відсотках).

Стратегія проектування дизайну сайтів для мобільних пристроїв:

1) визначити групи пристроїв за шириною екранів у пікселах;

2) вибрати пристрій, для проектування настроювань за замовчуванням (вони будуть застосовуватися у випадку, якщо розмір екрана пристрою не ідентифікований додатком або не був передбачений);

3) визначити правила адаптації для контенту й функціональних елементів: які елементи завжди повинні залишатися на екрані, які елементи можуть бути відкинуті, які можуть розтягуватися, за якими принципами відбувається зміна розмірів, які елементи не повинні мінятися;

4) використати можливостей HTML і CSS для проектування гнучкого дизайну, який здатен підлаштовуватися.

3. *Сенсорний екран.* При використанні пристроїв із клавіатурою й мишею елементами на відносно великому екрані управляють за допомогою невеликого

курсору миші або клавіатурних сполучень. Користувачам мобільних пристроїв із сенсорним екраном доводиться попадати пальцем, який значно крупніший за курсор миші, по дрібних деталях. Тому розмір посилань, кнопок й інших функціональних елементів повинен бути достатнім для зручної вказівки пальцем. Розмір елементу повинен бути як мінімум 1 сантиметр з кожного боку, й бути відділеним порожнім простором від сусідніх елементів. Звичайно, це можливо тільки за рахунок зменшення доступного простору для іншого контенту.

4. *Труднощі набору тексту.* Ніхто не любить друкувати за допомогою клавіатури на сенсорному екрані. Тому дизайн потрібно проектувати так, щоб виключити набір тексту або звести його до мінімуму. Для веб-додатків у випадку мобільних версій особливо актуальне використання автопідстанови закінчень (рис. 7.21). Тобто коли користувач починає заповнювати текстове поле, у випадаючому списку йому пропонується вибрати можливі варіанти закінчення фрази, яка набирається.

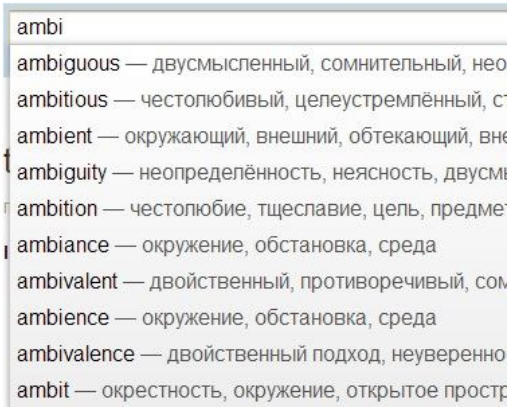


Рис. 7.21 Автопідставнова закінчень

Використовувати такий прийом ефективно при наборі користувачем адресів сайтів, дат, власного ім’я, назв файлів і т. п. Автопідстанова закінчень береже сили, час і когнітивний опір користувача.

5. *Мінливість навколишніх умов.* Люди користуються телефонами й іншими мобільними пристроями в найрізноманітніших умовах: на вулиці при яскравому сонці, у темному залі театру, у конференц-залі, машині, автобусі, поїзді, літаку, магазині, ванній кімнаті й навіть у ліжку. Цю різницю в

зовнішнім висвітленні потрібно враховувати при підборі кольорів. Наприклад, світло-сірий текст на темно-сірому фоні буде зовсім не видно при прямих сонячних променях.

Другий аспект – це розмаїтість зовнішніх шумів. У транспорті користувачі можуть і не почути звуків, які видає їх телефон, у той же час як у тихому громадському місці раптовий звук може здатися дратівним й недоречним. Тому звукові сигнали варто використовувати дуже акуратно.

І третій обтяжуючий зовнішній фактор – це рух. Читання дрібного тексту й натискання «товстим пальцем» проблематично навіть у самих ідеальних умовах, і стає просто нестерпно в стрибучому по купинах автобусі або вагоні потягу, який погойдується на рейках. І це ще один привід задуматися про підбір оптимальних розмірів функціональних елементів і тексту контенту.

6. *Соціальний вплив і обмежена увага.* У більшості випадків користувачі мобільних пристроїв не зосереджують всю свою увагу на додатку. Вони дивляться на інтерфейс додатку під час прогулянки, їзди на велосипеді, розмови з іншими людьми, проведення наради або навіть водіння автомобілю (хоча останнє й забороняється правилами дорожнього руху).

Буває, звичайно, що іноді користувачі повністю фокусуються на мобільному пристрої, наприклад під час ігор, але однаково цю концентрацію не можна порівнювати з концентрацією на такій самій грі, але при використанні стаціонарного комп'ютера з повноцінною клавіатурою у себе вдома.

Тому проектувати треба так, начебто користувачі дуже неуважні: сценарії користувацьких інтерфейсів повинні бути дуже прості, швидкі, дозволяти скасування дії й зрозумілі без додаткових пояснень.

Соціальний вплив полягає в тому, що при використанні мобільного пристрою людина часто перебуває в оточенні інших людей. В одній ситуації вона може захотіти показати що-небудь на екрані іншій людині. В іншій ситуації навпроти зволіє щоб ніхто сторонній не зміг побачити через плече що знаходиться на екрані.

Те ж саме стосується звуків – в одній компанії гучний пристрій буде заважати, в іншій хочеться поділитися новим записом із друзями.

Створити збалансований дизайн для всіх ситуацій непросте завдання.

Підходи до мобільного проектування.

1. Визначити, що дійсно необхідно користувачеві. Звичайне використання мобільних додатків пов'язане з тим, що треба терміново щось довідатися; є вільна хвилинка, треба її чимсь зайняти; потрібно з кимось зв'язатися; отримати інформацію щодо того місця, де користувач зараз перебуває.

2. Забрати із сайту або додатку все, крім його суті: зайвий контент, бічні панелі, оголошення, карти сайту, соціальні посилання, скоротити до мінімуму логотипи й т. п. Варто зосередитися на декількох завданнях, які мобільні користувачі бажають одержати від додатку й переконатися, що відповідний контент з'являється у головному вікні (головній сторінці) зверху.

Рис. 7.22 Порівняння сайтів NBA 2010 і 2013 років

На рис. 7.22 подано дві мобільні версії сайту ігор NBA. Про першу розповідається в книзі Дженніфер Тідвелл [15] як про приклад невдалого мобільного дизайну. Тому що логотип, реклама, оголошення й засоби навігації зайняли весь доступний простір, і властиво на інформацію про матчі й місця

залишалося лише 20% екрану. У новій версії інформація про ігри займає вже 80%. Скоротивши, таким чином, інтерфейс сайту, варто все-таки залишити можливість доступу до повного дизайну тим користувачам, які цього дійсно захочуть. Для цього треба розмістити посилання на повну версію в помітному місці. Відкидання всіх деталей може зробити сайт невпізнанним. Тому деяку версію навігації по структурі додатку можна залишити, полегшивши її за рахунок скорочення функцій, застосування більш вузьких шрифтів і т. п.

3. По можливості, варто використовувати апаратну частину мобільного пристрою. Мобільні пристрої пропонують можливості, які відсутні на робочому столі: датчик положення екрану (горизонтальне/вертикальне), камера, вбудований звук, введення даних жестами, відчутний зворотний зв'язок (наприклад, вібрація). Деякі пристрої є багатозадачними й дозволяють запускати додаток у фоні.

4. Лінеаризація контенту. Цей підхід пов'язаний із проблемою обмеження по ширині. Вертикальне розташування елементів підвищує читабельність додатку. Для лінеаризації контенту розроблений спеціальний патерн проектування під назвою *вертикальний стек*. Даний патерн припускає розташування змісту на сторінці у вигляді вертикальної колонки, з виключенням або значним обмеженням розташування елементів пліч-о-пліч. Текстові елементи розділяються лінією, сторінка прокручується до самої глибини екрану, яку дозволяє пристрій. Більшість мобільних веб-сторінок, які передбачається використовувати на пристроях різних розмірів повинні застосовувати цей патерн, особливо якщо вони містять текст (основний контент і форми). До імерсивного контенту (контенту, у якому можуть бути реалізовані будь-які форми зв'язку з людиною) вертикальний стек не застосовується – такий контент не потрібно прокручувати по вертикалі. Прикладами імерсивного інтерфейсу в даному контексті може служити відео на повний екран, ігри.

Патерн вертикальний стек кращий саме для веб-сторінок, тому що перехід між окремими сторінками вимагає часу на завантаження кожної. У випадку ж

додатків установлених на самому пристрої, коли перехід між сторінками є моментальним краще розташовувати вміст на одиночних екранах, які не прокручуються вертикально.

7. Оптимізація взаємодії

Після виділення основних завдань користувача й відповідного скорочення функцій і контента додатку, потрібно зробити ці функції максимально простими в управлінні. Це можливо за рахунок застосування наступних рекомендацій:

- відмова від набору тексту користувачем або скорочення по можливості до декількох символів;
- зменшення кількості сторінок веб-додатків настільки, наскільки це можливо. Так само варто не перевантажувати сторінки зайвими байтами. Невідомо де буде людина користуватися додатком, і яка якість зв'язку там буде;
- скорочення бічного прокручування, крім випадків, коли це дозволяє позбутися завантаження сторінок і введення символів. Іншими словами, у випадку, якщо треба представити велику кількість контенту, між довгими вертикальними сторінками й більшою кількістю маленьких сторінок варто вибирати перше;
- скорочення кількості натискань на функціональні елементи при пошуку користувачем інформації, яка його цікавить, або при виконанні завдання. Звісно натискання на великі цілі або використання апаратних кнопок краще, ніж набір довгих послідовностей символів, але будь-який вид натискань також потрібно зменшувати.

Тема 8. **Оцінка якості людино-машинного інтерфейсу. Юзабіліті-тестування.**

8.1. Кількісна оцінка інтерфейсу.

Кількісні методи допомагають звести суперечливі питання в оцінці якості інтерфейсу до простих обчислень.

Одним із кращих підходів до кількісного аналізу моделей інтерфейсів є класична модель *GOMS (the model of goals, objects, methods and selection rules)* – модель цілей, об'єктів, методів і вибору правил.

Ця модель заснована на оцінці швидкості печатання. Час, необхідний для виконання якогось завдання системою користувач-комп'ютер, є сумою всіх часових інтервалів, які були потрібні системі на виконання елементарних жестів, з котрих складається дане завдання.

Лабораторним шляхом установлені *стандартні середні інтервали* для деяких жестів, виконуваних різними користувачами:

K = 0,2 с, – натискання клавіші;

P = 1,1 с, – вказування (на якусь позицію на екрані монітора);

H = 0,4 с, – переміщення (руки із клавіатури на ГПВ або з ГПВ на клавіатуру;

M = 1,35 с, – ментальна підготовка;

R – відповідь (час очікування відповіді від комп'ютеру).

Розрахунок за моделлю GOMS.

Основні правила, що дозволяють визначити, у які моменти будуть проходити ментальні операції, представлені в табл. 4.

Таблиця 4. Правила обліку ментальних операцій при кількісній оцінці інтерфейсу

Правило 0. Початкове розміщення операторів *M*.	Оператор *M* встановлюється перед всіма операторами *K* (натискання клавіші) і *P* (вказування), призначеними для вибору команд, якщо P указує на аргументи цих команд оператор *M* не ставиться.
Правило 1. Видалення очікуваних операторів *M*.	Якщо оператор, наступний за оператором *M* є очікуваним з погляду оператора, що передує *M*, то цей оператор *M* може бути вилучений й послідовність *РМК* перетворюється у *РК*.

Правило 2. Видалення операторів *M* усередині когнітивних одиниць.	Якщо рядок типу *МКМКМК*…належить когнітивній одиниці, те варто видалити всі оператори *M*, крім першого. Когнітивною одиницею є безперервна послідовність символів, що вводяться, які можуть утворювати назву команди або аргумент.
Правило 3 Видалення М перед послідовними роздільниками.	Якщо оператор *K* означає зайвий роздільник, який стоїть наприкінці когнітивної одиниці (наприклад роздільник команди, яка проходить відразу за роздільником аргументу цієї команди) то варто видалити оператор *M*, який розташований перед ним.
Правило 4 Видалення операторів *M*, які є переривниками команд.	Якщо оператор *K* є роздільником, який стоїть після постійного рядка (наприклад, назва команди або будь-яка послідовність символів, що щораз уводиться в незмінному вигляді), то варто видалити оператор *M*, який стоїть перед ним. Але якщо оператор *K* є роздільником для рядка аргументів або будь-якого іншого змінюваного рядка, те *M* варто зберегти перед ним.
Правило 5 Видалення операторів, що перекриваються, *M*.	Будь-яку частину операторів *M*, що перекривають оператор *R*, який означає затримку, пов'язану з очікуванням відповіді комп'ютеру, враховувати не слід.

8.2. Якісна оцінка інтерфейсу

Час виконання сценаріїв інтерфейсу є тільки непрямим показником зручності його використання. Провести якісну оцінку інтерфейсу можливо тільки за допомогою користувачів.

У *методі фокусних груп* невелика група людей (зазвичай від 5 до 8 чоловік) сідає за стіл й оцінює ідеї й зразки дизайну, які їм демонструються. Це груповий процес, і його результати засновані головним чином на тому, як учасники групи реагують на думки один одного. Метод фокусних груп підходить для того, щоб швидко одержати діапазон думок і оцінок користувачів із приводу тих або інших речей. Він скоріше підходить для того, щоб зібрати ідеї для розробки дизайну інтерфейсу.

При *юзабіліті-тестуванні* (тестування на зручність використання) кожному користувачеві окремо демонструється дещо (це може бути готовий

продукт, спробний зразок продукту або макети окремих сторінок), і потім його просять або спробувати зрозуміти, що він бачить, або виконати деяке завдання.

Для проведення юзабіліті-тестування потрібно вирішити кілька питань:

1. Скільки разів потрібно тестувати?

Проводити тестування необхідно на кожному етапі розробки або після внесення кожної зміни. Звичайно, це не завжди можливо. Але головне те, що навіть одне тестування краще, ніж жодного.

2. Скільки учасників вибрати?

Звичайно чим більше, тим краще. Але провести тестування на сотні людей можливо, тільки якщо тестується сайт, який вже має кілька сотень відвідувачів. Треба відзначити, що в цьому випадку оцінка результатів є більш кількісною ніж якісною, й потребує статистичних методів обробки. Успіх тестування – це величина багатофакторна. Дослідження показали, що більше проблем буде виявлено, якщо провести 2 тестування із трьома учасниками, ніж одне тестування з вісьма.

3. Ким повинні бути учасники тестування?

Говорять, що в тестування обов'язково повинні брати участь потенційні користувачі. Добре якщо це можливо, але в цілому значення цільового користувача часто перебільшують. Головне правило, що учасники тестування не повинні бути в числі розробників проекту.

4. Якими методами тестувати?

Перш ніж проводити тестування потрібно визначитися з наступними моментами:

а) що буде тестуватися?

б) яка мета тестування, що потрібно довідатися в результаті?

в) де буде походити тестування й скільки часу це забере?

г) які завдання повинні будуть виконати учасники тестування?

д) як буде одержано від них об'єктивне відкликання?

Учасникам тестування важливо дати зрозуміти, що тестується додаток, а не вони, інакше учасники будуть боятися зробити помилку, а коли це

відбудеться можуть засмутитися, й почати робити помилки більше пов'язані з їхнім емоційним станом, чим з дизайном додатку. Під впливом стресу людина не здатна адекватно сприймати інформацію на екрані й схильна повторювати ту саму дію знову й знову, навіть якщо вона не приводить до бажаного результату. Якщо ж відомо заздалегідь, що в більшості випадків використання продукту буде відбуватися в стресових ситуаціях, необхідно визначити можливий рівень стресу й у відповідності з цим змінити дизайн.

Метою тестування може бути визначення:

- кількості помилок;
- частоти повторюваності помилок;
- що саме викликає питання в користувача й вимагає пояснень;
- скільки часу потрібно на виконання тієї або іншої операції.

Часто метою може бути порівняння двох дизайнів за певними критеріями.

Однією із цілей тестування є перевірка того, як стресова ситуація може вплинути не поводження користувача. Не варто припускати, що люди будуть використовувати продукт винятково в спокійній обстановці. Ті речі, які розробнику не здаються сприятливими виникненню стресу, можуть викликати стрес у тих, хто буде використовувати продукт у реальному світі. Процес зборки іграшки для дитини опівночі напередодні її дня народження викликає стрес. Заповнення форми на екрані, коли клієнт «висить» на іншому кінці дроту або є присутнім особисто, викликає стрес. Більшість ситуацій пов'язаних з медициною викликають стрес. Оплата рахунків тим більше викликає стрес.

Закон Йеркса-Додсона затверджує, що здатність до виконання завдання зростає під впливом фізіологічного або розумового стимулу, але тільки до певної точки. Коли стимул стає занадто більшим, ця здатність зменшується – увага розсіюється, люди впадають у занепокоєння, здатність до розв’язку проблем знижується й виникає *тунельний ефект*. Користувач повторює дії знову й знову, хоча вони не працюють. Проведення досліджень на предмет, як користувачі справляються зі стресовими ситуаціями в лабораторних умовах, проблематично. Це можна визначити вже на етапі використання продукту.

Добре створити спеціальний сайт для відкликань й інтерв'ю з людьми, які використовують конкретний продукт, або форум, де вони будуть обговорювати проблеми. Так у системі онлайн навчання Coursera на сайті курсу з «Людино-машинний інтерфейс» є форум https://class.coursera.org/hci-2012-002/forum/index, на якому учасники обговорюють все що їм незрозуміло, діляться порадами й труднощами. Адміністрація курсу майже не бере участь у переписці, тільки іноді залишає повідомлення про зміну у структурі й плані курсу. Ці рішення адміністрація приймає на основі висловлюваних на форумі думок. Наприклад, якщо більшість студентів не встигають зробити завдання в зазначений термін, то строки переносяться.

Для визначення того, як користувачі справляються із завданнями, використовуються наступні методи:

- *міркування вголос* (учасник повністю коментує все, що він бачить, робить або збирається робити в процесі виконання завдання);
- *спостереження* (той, хто проводить тестування дивиться з боку й фіксує за допомогою блокнота й ручки, фото- і відеоапаратури все, що робить учасник тестування);
- *постінтерв'ю* (після виконання завдання учасник у режимі живої бесіди або письмових відповідей, у заздалегідь підготовлених питальниках, розповідає, що йому сподобалося, не сподобалося, які труднощі виникли).

При складанні питань для інтерв'ю варто уникати таких, що вимагають відповіді так чи ні. Наприклад – вам сподобався мій дизайн? Це корисний інтерфейс? Або питань, які містять відповідь у собі, наприклад – як сильно вам сподобався мій дизайн? Питання повинно бути відкритим, щоб одержати від учасника тестування розгорнуту відповідь. Якщо користувач не зможе описати призначення елементів дизайну однією короткою пропозицією й назвати всі найважливіші об'єкти через пару днів без підказок, то вибір цих об'єктів невдалий.

Що стосується інтерв'ю, то його результати найбільш адекватні при особистому спілкування. Дослідження Чарльза Ніквіна з Університету де Пола

показали, що люди найбільше брешуть у телефонній розмові, й найменше на папері. Люди озиваються про інших більш негативно в повідомленнях електронної пошти, а не на папері, тому при проведенні опитування електронною поштою, варто враховувати, що відповіді будуть більше негативними й менш правдивими.

5. Як інтерпретувати результати?

Зібрана під час юзабіліті-тестування інформація повинна допомогти внести такі зміни в дизайн, які знизять імовірності здійснення помилок користувачами. Тому всі знайдені помилки варто задокументувати, розібрати за категоріями і відзначити наслідки кожної.

Помилки з позитивними наслідками – це дії, які не ведуть до бажаного результату, але надають людині інформацію, за допомогою якої вона досягає більш загальної мети. Наприклад, у пошуках однієї функції додатку користувач випадково відкрив іншу, яка буде йому потрібна наступного разу. *Помилки з негативними наслідками* заводять у тупик, знищують позитивні результати, повертають людину у вихідну точку або приводять до непоправних наслідків. *Помилки нейтральні* не впливають на виконання завдання – користувач зробив помилку, сам виправив й продовжив роботу.

При перепроектування неможливо виправити всі помилки, але в першу чергу варто виключити ті, що приводять до негативних наслідків.

Категорії помилок можна представити як: *помилки виконання* – помилки, які допускаються при проходженні кроків процедури (зайві дії, пропущена дії, неправильні дії); *помилки управління* – при управлінні пристроями (наприклад замість того, щоб повернути картинку на планшеті користувач переходе до наступної).

Ще одним важливим моментом, на який варто звернути увагу при тестуванні – які стратегії виправлення помилок застосовують користувачі. Цільова аудиторія може використовувати різні стратегії для виправлення помилок. Не варто припускати, що люди старшого покоління виявляться не в змозі впоратися із завданням. Вони, можливо, застосують інший підхід, і це

може зайняти більше часу, але вони здатні вирішувати ті ж завдання, що й молоді. До того, крім розходжень у віці, існують ще й новачки й майстри.

Основні стратегії виправлення помилок.

Системний підхід. Люди планують процедури, які будуть використані для усунення помилок. Намагаючись зрозуміти, як включити циклічне відтворення пісні або музичної композиції на своєму планшеті, користувачі намагаються обійтися одним меню. Коли це не спрацьовує, вони перевіряють кожний пункт.

Метод проб і помилок. У відмінності від системного підходу, метод проб і помилок припускає, що дії, меню, значки й способи керування вибираються випадковим чином.

Жорстке дослідження. Повторення однієї й тієї ж дії знову й знову, хоча вона не приводить до бажаного результату.

Дані зібрані за кожною категорією помилок, дані про застосовані користувачами стратегії виправлення помилок допоможуть змінити дизайн

Принципи відбору проблем:

- не звертати увагу на проблеми «каяку» (випадкові помилки, які користувач відразу сам виправив);
- не піддаватися спокусі додавати дещо;
- ставитися скептично до прохань користувачів про додавання нових сервісів;
- виправляти очевидні помилки.

Тема 9. **Засоби розробки людино-машинного інтерфейсу**

Існує безліч засобів для розробки користувацького інтерфейсу. Це й засоби для дизайнерів не знайомих із програмуванням (Photoshop, Corel, 3DMax і т.д.), для програмістів, які не є дизайнерами (додаткові бібліотеки примітивів, компонентів інтерфейсу у звичних середовищах розробки Visual Studio, Delphi, NetBeans), для дизайнерів, знайомих з мовами програмування, що дозволяють специфікувати функції вводу-виводу, а також визначати, застосовуючи техніку безпосереднього маніпулювання, елементи інтерфейсу.

Інформаційний інструмент для розробок на ранніх етапах дизайну користувацьких інтерфейсів веб-сайтів запропонували у університеті Вашингтона за назвою DUB – DENIM. Додаток є доступним на сайті університету. Додаток замінює папір і ручку, тому його рекомендується використовувати для планшету зі стіліусом.

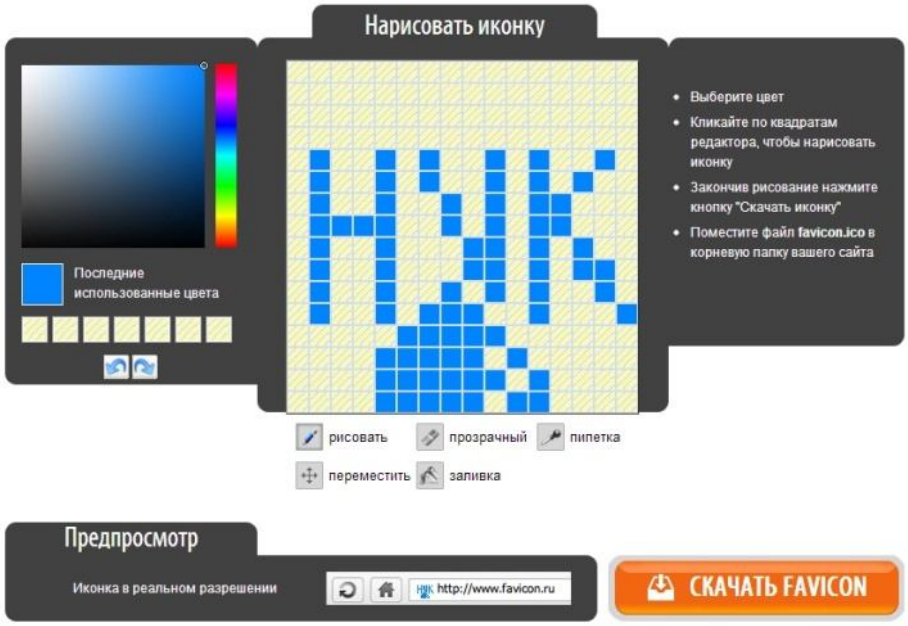


Рис. 9.1 Створення іконок для сайтів на favicon.ru

Професійні дизайнери віддають перевагу графічному редактору Adobe Photoshop, який дозволяє створювати гнучкі прототипи завдяки шарам. Можна створити самостійно основні функціональні елементи на окремих шарах, і використовувати їх у різних проектах, або скористатися вже готовими наборами у форматі *.psd, такими як Small GUI Pack, iPad GUI Set (набір елементів, які можна редагувати, для iPad) , Turquose PSD (шаблони для Windows) ALL In One Web Elements Kit (елементи для веб-сайтів), OSX Leopard GUI Set (шаблони FhotoShop, які редагуються з шаблонами в стилі Mac OS), Android UI Elements Set.

Веб-додаток Android Asset Studio – цей інструмент допоможе в створенні іконок, які можна використати в будь-якому додатку Android. Для кожного елементу задаються основні параметри (кольори, розміри) і на тій самій

сторінці можна відразу побачити результат. Готовий елемент у форматі *.png і в трьох різних дозволах можна завантажити на свій комп'ютер.

На favicon.ru можна створювати іконки для сайтів попікселно. Також Дозволяється автоматично міняти дозвіл і завантажувати результат на свій комп'ютер.

Для побудови начерків дизайну інтерфейсу на початкових етапах прототипування існують прості у вивченні безкоштовні веб-ресурси, які дозволяють вирішити задачу розміщення основних елементів, не зосереджуючись на деталях. Серед них варто відзначити balsamiq.com, diagram.ly, caccoo.com. Flex 3 Stensil надає більш широкий вибір елементів інтерфейсу, але вимагає установки.

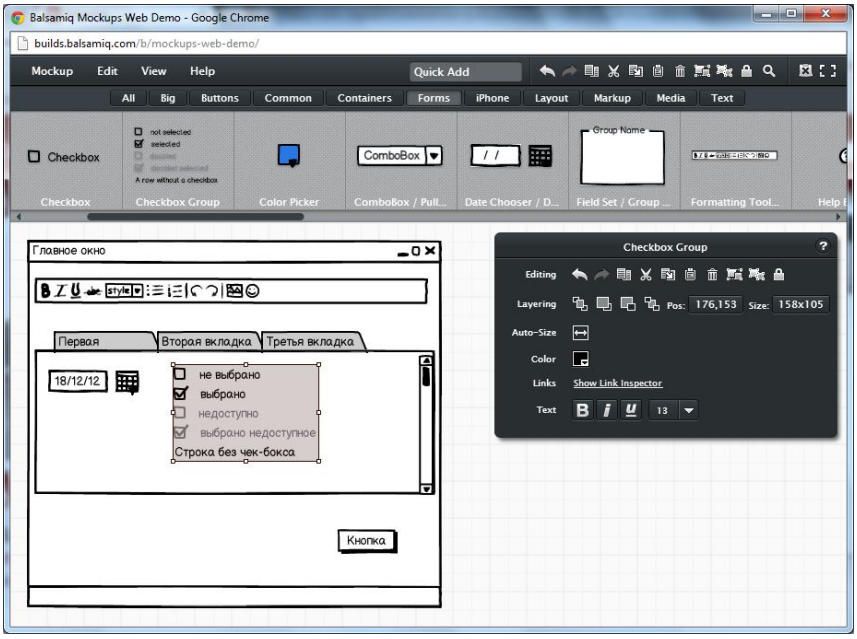


Рис. 9.2 Робоча область інтернет-версії редактора макетів Balsamiq

На рис. 9.2 показане моделювання вікна додатку в середовищі Balsamiq. Панель інструментів Balsamiq складається з тематичних вкладок, які містять елементи інтерфейсів відповідних типів. У поданому випадку активна вкладка Forms і видимі елементи, характерні для прикладних додатків віконного типу. Але є можливість проектування додатків для смартфонів, веб-додатків і т. п. Всі елементи виконані схематично, вони начебто намальовані від руки. Це зроблено цілеспрямовано, щоб увага розробника була зосереджена на

позиціюванні елементів з погляду зручності виконання користувальницьких сценаріїв.

Найбільше широко подібні ресурси представлені для проектування сайтів. Серед них User Interface Design Framework, Hot Gloo.

Наступна група засобів дозволяє створювати діючі прототипи, які вже дозволяють одержувати зворотний зв'язок. Lumzy являє собою інструмент для створення начерків і «риби» веб-сайтів і додатків. За допомогою даного засобу, можна розробляти начерки й відсилати їх своїм клієнтам. Додаток також підтримує командний процес розробки, містить у собі чат для обговорення дизайнів і начерків, вказівок версії файлів і багато чого іншого.

Для проектування інтерфейсів веб-додатків можна скористатися конструкторами сайтів (ucoz.ua, weebly.com), які надають безкоштовний хостінг, професійні теми, мультимедійні засоби.

Mocklinkr допоможе подати статичні макети у вигляді повноцінних кликабельних веб-сайтів. MockFlow – це онлайн додаток для розробки начерків користувацьких інтерфейсів не тільки для веб-сайтів, але й для декстопних додатків.

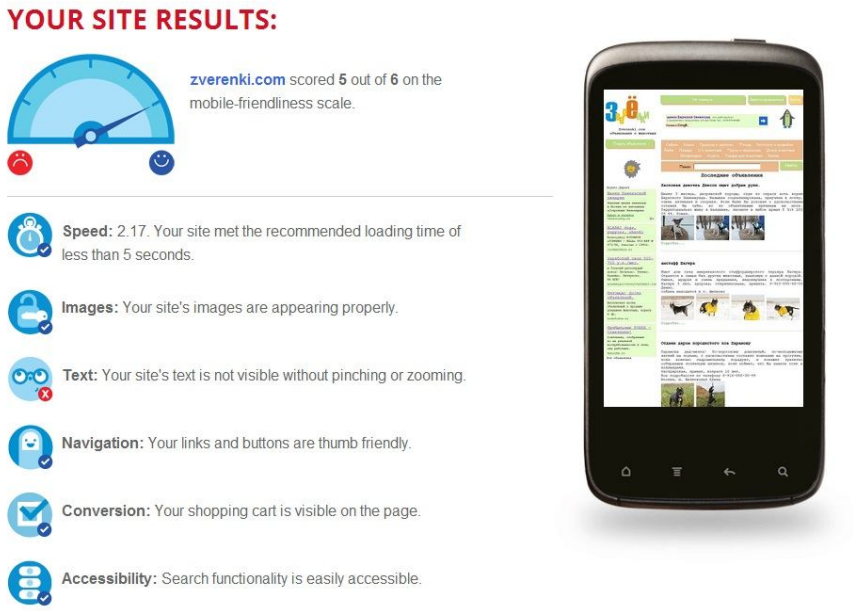


Рис. 9.3 GoMoMeter від Google. Оцінка дружності сайту

Багато бібліотек і шаблонів створено для javaScript-розробок:

- DHTMLX являє собою js-бібліотеку, яка надає можливість розробки кроссбраузерних користувацьких інтерфейсів на Ajax. Бібліотека виконана у двох версіях – безкоштовній й комерційній. Модульна архітектура бібліотеки дозволяє використати окремі блоки компонентів і комбінувати їх у загальний інтерфейс на основі Ajax;
- LivePipe UI – набір високоякісних віджетів і функціональних елементів для web 2.0 додатків, створених на основі Prototype JavaScript Framework;
- jQTouch jQuery – плагін для розробки мобільних додатків для iPhone, iPod Touch і інших інноваційних пристроїв;
- Uki – це швидкий js-набір інструментів для розробки веб-додатків і додатків на основі робочого стола. Набір надає багату бібліотеку компонентів;
- MochaUI – це бібліотека користувацьких інтерфейсів веб-додатків, розроблених за допомогою платформи Mootools.

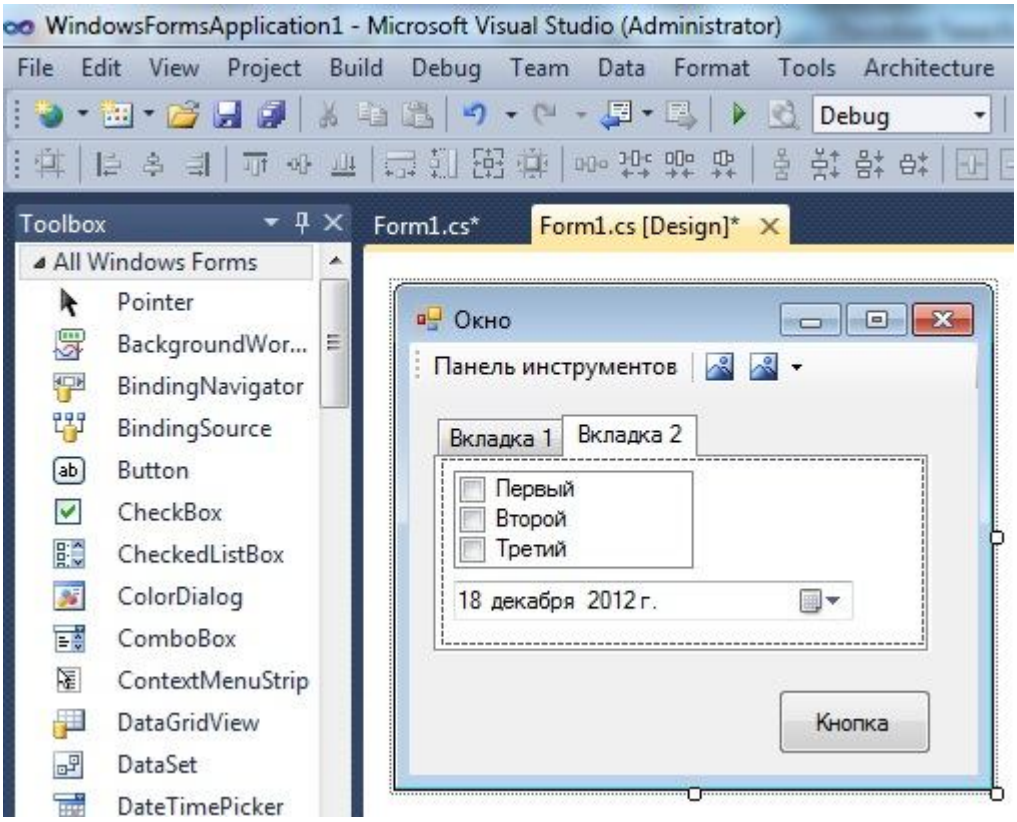


Рис. 9.4 Робоча область Microsoft Visual Studio для створення WinForms

Готові сайти варто перевіряти з приводу, як вони будуть виглядати на мобільних пристроях. Нижче наведено кілька безкоштовних веб-додатків, які здійснюють подібну перевірку:

- GoMoMeter (рис. 9.3) – сервіс від Google, за допомогою якого можна побачити, як сайт виглядає на смартфоні, і отримати безкоштовний звіт з персональними рекомендаціями.
- iPhone Tester – емулятор iPhone, який дозволяє перевірити сайт на екранах iPhone 3G і 4G, і IPod Touch. Для перевірки досить увести адресу сайту в адресний рядок.
- Mobile Phone Emulator – емулятор, який дозволяє побачити сайт на екранах різних мобільних телефонів, включаючи iPhone, HTC, LG, BlackBerry і Samsung.

Високорівневі засоби розробки програмних продуктів мають бібліотеки візуального проектування інтерфейсів. Це вже згадані раніше Microsoft Visual Studio (створення додатків на C# і C++), Delphi, NetBeans (створення додатків на Java).

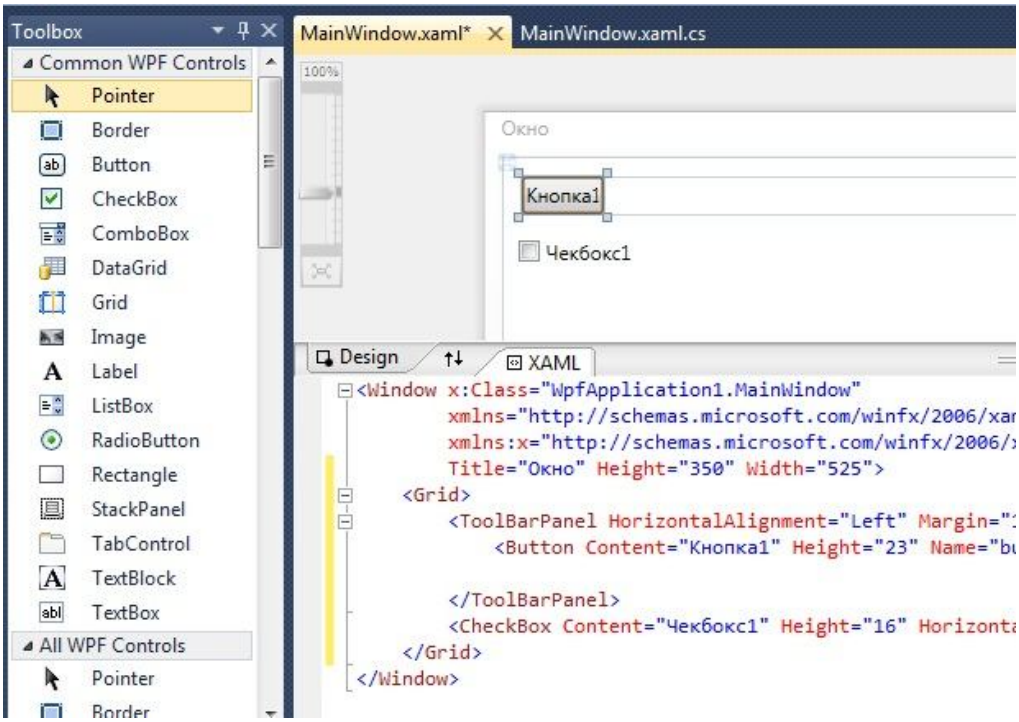


Рис. 9.5 Робоча область Microsoft Visual Studio для створення інтерфейсів на основі WPF

Microsoft Visual Studio дозволяє проектування веб-интерфейсів на HTML+CSS, а так само проектування додатків на основі робочого стола з використанням звичних WinForms для створення віконного інтерфейсу (рис. 9.4), на основі WPF (windows presentation foundation) (рис. 9.5). В основі WPF лежить векторна система візуалізації, яка не залежить від дозволу пристрою виводу, й створена з урахуванням можливостей сучасного графічного встаткування. WPF надає засоби для створення візуального інтерфейсу, включаючи мову XAML (Extensible Application Markup Language – мова на основі XML), елементи управління, прив'язку даних, макети, двомірну й тривимірну графіку, анімацію, стилі, шаблони, документи, текст, мультимедіа й оформлення.

Частина 2. ПРАКТИКУМ ЗА КУРСОМ "ЛЮДИНО-МАШИННА ВЗАЄМОДІЯ"

Практичні завдання для закріплення теоретичного матеріалу з прикладами виконання

Практичне завдання 1

Запропоновано теми для розробки інтерфейсів програмного забезпечення й рекомендований список персонажів за кожною темою.

Завдання:

1. Доповнити опис персонажів (деталі про звички, хобі, рівень користування комп'ютерною технікою, бажання вчити нові програми);
2. Написати цілі кожного персонажу;
3. Написати функції необхідні кожному персонажу;
4. Вибрати ключовий персонаж для розробки інтерфейсу;
5. Визначити основні функції додатку;
6. Визначити додаткові можливості додатку.

Теми:

- календар-планувальник студента (розклад занять, розклад тренувань, строки здачі завдань, кількість зданих завдань і тих, що залишилися).

Рекомендований список персонажів:

  - Таня, лінгвістика, 2 курс;
  - Петро, зварювання, 1 курс;
  - Микола, програмування 5 курс;
  - Олена, економіка 3 курс.

- планувальник сімейного бюджету (облік доходів і витрат, облік накопичень готівкою й безготівкових, планування покупок).

Зразковий список персонажів:

- Іван Іванович, токар 1 розряд, пенсіонер;
- Надія, домогосподарка, швачка за освітою, 30 років;
- Ганна Петрівна, бухгалтер, 40 років;
- Віталій Андрійович, підприємець, власник мережі торговельних точок, 35 років.

- журнал викладача (облік відвідувань студентів за групами і предметами, контроль зданих робіт, контроль залікових оцінок).

Рекомендований список персонажів:

- Роман Васильович, професор, механіка корабля, 70 років;
- Владислав Львович, доцент, інформаційні технології, 40 років;
- Олена Сергіївна, асистент, лінгвістика, 25 років.

- текстовий редактор для блогів (простий, швидкий).

Рекомендований список персонажів:

- Настасія Володимирівна, ветеринар, 45 років;
- Антон, програміст, 30 років;
- Маргарита Дмитрівна, вихователь дитячого садка, 60 років;
- Настя, школярка, 13 років.

- ПЗ для туристичної фірми.

Рекомендований список персонажів:

- Наталя, директор, філолог за освітою, 30 років. Укладає договори з туроператорами, наймає співробітників, продає тури.
- Антонина Пантелєєвна, бухгалтер, 50 років. Веде облік доходів/витрат, здійснює нарахування зарплати, відрахування в податкову інспекцію.
- Світлана, стажист, студентка спеціальності менеджмент туризму, 20 років. Розповідає про тури клієнтам, показує фото, бронює тури, продає тури.

- сайт – дошка оголошень для продажу авто.

Рекомендований список персонажів:

- Іван Іванович, токар 1 розряд, пенсіонер;
- Антон, програміст, 30 років;
- Маргарита Дмитрівна, вихователь дитячого садка, 60 років;
- Віталій Андрійович, підприємець, власник мережі торговельних точок, 35 років.

Приклад профілю користувача програмного забезпечення для ПЗ домашнього сканера.

Іван

Соціальні характеристики:

Власник маленької фірми по створенню сайтів.

Мотиваційне середовище:

Не бажає розбиратися в дозволах, сканерах, настроюваннях.

Ціль:

Заробити гроші шляхом створення сайтів

Навички й уміння:

Він не розбирається глибоко в технічних питаннях і не є дизайнером, однак знайомий з комп'ютерами й знає, що оптимізовані зображення набагато швидше скачуються з Інтернету, ніж великі, повнокольорові.

Вимоги до ПЗ:

Одержати зображення середнього дозволу для сайтів, без зайвих витрат і не вдаючись у деталі процесу.

Завдання користувача:

- швидко знайти відскановане зображення;
- обрізати зображення;
- легко вставити в дизайн сайту за допомогою програм проектування сайтів.

Робітниче середовище:

Операційна система Linux

Практичне завдання 2

Для поданих тематичних напрямків виділити по три профілі користувачів. Описати профілі, вибрати ключовий для розробки інтерфейсу. Пояснити вибір.

Рекомендовані теми ПО:

- банкомат;
- інтернет-магазин «Полювання та риболовля»;
- навігатор для таксистів;
- обчислення визначених інтегралів;
- конструктор сайтів;
- електронний довідник автобусних сполучень;
- бронювання готельних номерів;
- сервіс електронних листівок;
- електронне тестування з математики.

Практичне завдання 3

Для заданих ситуацій розробити користувацькі сценарії використання відповідного програмного забезпечення:

1. Віктор (чоловік 30-ти років) домовився зустрітися із другом у торгівельному центрі. Друг затримується вже на півгодини. Віктор хотів подзвонити другові, але виявив, що на рахунку недостатньо грошей. Поруч із місцем зустрічі стоїть термінал поповнення. Зазвичай Віктор поповнює телефон скретч-картками, але зараз він не хоче відходити від місця зустрічі й вирішує поповнити рахунок через термінал. Надати сценарій використання програмного забезпечення для поповнення рахунку мобільного телефону через термінал.

2. Студентка Таня приїхала до Києва з невеликого міста оглянути визначні пам'ятки самостійно. Пройшовши Хрещатиком, вона вирішила пішки дістатися Андріївського узвозу. Удома Таня вивчала карту й по пам'яті знайшла дорогу до пам'ятника Богдану Хмельницькому. Але далі розгубилася, куди повернути. У Тані є смартфон з діагоналлю екрану 5 дюймів.

Запропонувати сценарій використання електронних карт для мобільних телефонів з метою пошуку певної вулиці від заданого місцерозташування.

3. Семен Іванович працює в українському культурному центрі керівником тріо бандуристів. Його відправляють у відрядження по роботі у Львів на тиждень для обміну досвідом в аналогічний центр. До цього часу Семен Іванович Львів ніколи не відвідував. Йому треба знайти бюджетне житло близько до цілі відрядження, а також, щоб було зручно від житла дістатися залізничного вокзалу й центру міста для огляду визначних пам'яток. Семен Іванович дуже боїться загубитися. Крім усього іншого, Семенові Івановичу потрібно буде відзвітуватися в бухгалтерії культурно центру, тому треба отримати квитанцію на розходи за проживання.. Надати користувацький сценарій використання інтернет-сервісу для пошуку житла в іншому місті.

4. Учителька математики Мар’я Василівна, за сумісництвом молодий і прогресивний класний керівник 5-Б, везе переможців обласної олімпіади з математики й їхніх батьків, загальною кількістю 10 чоловік, на всеукраїнську олімпіаду. Її задача замовити всім квитки бажано в одному плацкартному вагоні, не на бічних полках. Дві полки обов'язково повинні бути нижніми. Щоб впоратися із завданням спокійно й без поспіху, вона вирішує скористатися сервісом електронних квитків. Подати сценарій використання сервісу для замовлення квитків на групу людей.

5. Інженеру-кораблебудівнику на пенсії Марфі Петрівні, в якої є люблячий син, дуже подобається ходити в супермаркети й скуповувати все, що попадається на очі. Тому, незважаючи на любов сина, грошей на касі їй зазвичай не вистачає. Син подарував їй планшет з додатком, який дозволяє підраховувати вартість покупок в міру наповнення візка. Представити сценарій використання такого додатку в умовах супермаркету.

Приклад користувальницького сценарію.

Ситуація: використання банкомату. Операція: Видача грошей при перевищенні денного ліміту на суму грошей.

Користувач вставляє картку, в запропоноване поле уводить пароль, вибирає операцію «видача грошей», уводить суму, одержує повідомлення про перевищення денного ліміту й інформацію про доступну суму для отримання; змінює мову, змінює суму, одержує пропозицію забрати гроші після повернення карти, забирає карту, забирає гроші.

Практичне завдання 4

Оцінити зручність читання запропонованого тексту. Запропонувати скорочений варіант даного тексту й оцінити зручність його читання за формулами Флешу й Флешу-Кінкейду. Порівняти результати для тексту до й після внесених змін.

1. Текст із головної сторінки сайту:

*«Ваш профіль на форумі B**** і на цьому сайті – той самий. Додаткова реєстрація не потрібна.*

Розмістіть свою фотографію у вашому профілі. Як це зробити написане в інструкції Як розмістити фотографію в своєму профілі.

Укажіть більш інформації про себе. Це можна зробити у Особистому розділі. Прохання місто проживання писати англійською, тому що це один з параметрів за яком робиться вибірка при пошуку знайомств.

Бажано заповнити розділ Карта у вашому профілі, зайшовши в Особистий розділ. Там відзначити відвідані вами країни в списку країн галочками, що відобразиться на результуючій карті у вашому профілі.»

2. Текст головної сторінки сайту:

«Ви хочете змінити звичне "дім – робота – дім" на щось інше… Або на носі довгоочікувана відпустка, треба кудись поїхати, але їхати нема з ким. Є маса причин, щоб плюнути на звичне й виїхати в подорож. Добре, якщо є

хтось, хто вільний і підтримує ваше бажання подорожувати. А якщо немає? Тоді вам потрібний попутник. Сайт пошук попутників покликаний вам у цьому допомогти. Адже успіх або провал будь-якої подорожі залежать від правильної компанії, від попутника. Звичайно можна поїхати одному. Платити подвійно за номер в готелі або таксі...Намагатися познайомитися з людьми на місці... Але це не завжди виходить. Та й про людину краще довідатися заздалегідь...

Користуйтеся коментарями до події запрошення в попутники – пишіть ваші питання до можливого попутника. Обмінюйтеся там своїми фотографіями. Записуйтеся в попутники.»

3. Сайт інтернет-магазину. Правила повернення товарів:

«У частині повернення й обміну товарів ми дотримуємося Закону України "Про захист прав споживачів". Ми готові прийняти товар назад протягом 14 днів, якщо він зберіг товарний вид, цілісність упакування, всі свої технічні і якісні характеристики.

У випадку браку Ви також можете повернути або обміняти товар. Для повернення товару покупець повинен мати товарний чек, який свідчить про придбання даного товару в нашому магазині. Товар, придбаний в одному з наших роздрібних магазинів, можна повернути тільки у відповідний магазин.»

4. Довідка графічного редактору:

«***Перегляд зображення***

Зміна подання дозволяє вибрати спосіб роботи із зображенням. За бажанням можна збільшити певну частину зображення або все зображення повністю. І навпаки, якщо зображення занадто велике, його можна зменшити. Крім

того, у програмі Paint можна вивести лінійку й сітку, які підвищують ефективність роботи.»

Практичне завдання 5

Змінити запропоновані інтерфейси додатків, щоб скоротилася кількість дій при виконанні користувальницьких сценаріїв:

1. Вибір мови.

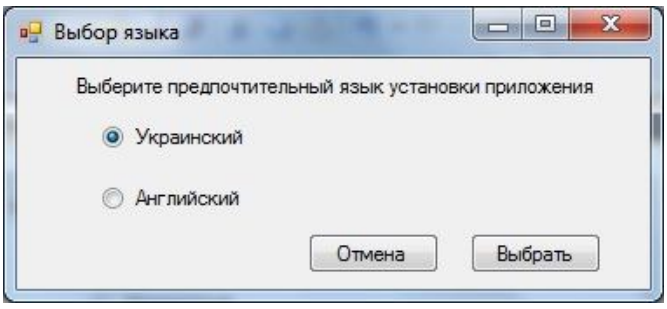


Рис. П1 Діалог вибору мови

2. Додавання коментарю до зображення.

Рис. П2 Додавання коментарю до картинки

3. Пошук слова в документі у випадку, якщо слово не знайдене.

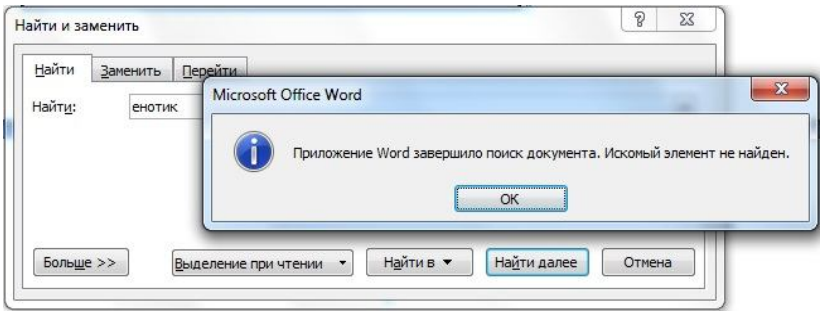


Рис. П3 Пошук слова в документі

Практичне завдання 6

Для сайтів, обраних випадковим чином провести оцінку інформативності логотипу, зробити висновки з:

- наявності рисунку – зв’язок з назвою й призначенням сайту;
- наявності назви сайту у логотипі;
- наявності слогану або пояснювального надпису, відображення у салогані напрямків діяльності сайту;
- естетичної оцінки слогану – стиль, кольори, розмір, погодженість з загальним дизайном сторінки сайту.

Практичне завдання 7

Запропонувати три варіанти ескізів логотипу для сайтів:

- інтернет-магазин подарунків;
- універсальна дошка оголошень;
- візитка розплідника морських свинок;
- візитка фірми з ремонту й продажу комп'ютерів;
- Microsoft;
- кулінарні блоги;
- університет;
- Верховна Рада.

Практичне завдання 8

Намалювати ескіз інтерфейсу існуючого додатку, поліпшивши його. Пояснити зроблені зміни. Як варіанти завдань можна взяти сайти, проаналізовані на якість веб-навігації при виконанні практичного завдання 7.

Практичне завдання 9

Провести кількісну оцінку наведених інтерфейсів при виконанні зазначених сценаріїв:

1. Вивчення нових слів. Слова подаються автоматично у випадковому порядку.

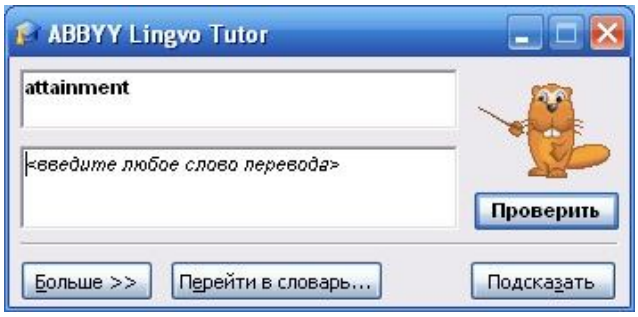


Рис. П4 ABBYY Lingvo. Навчання новим словам

2. Пошук перекладу слова.

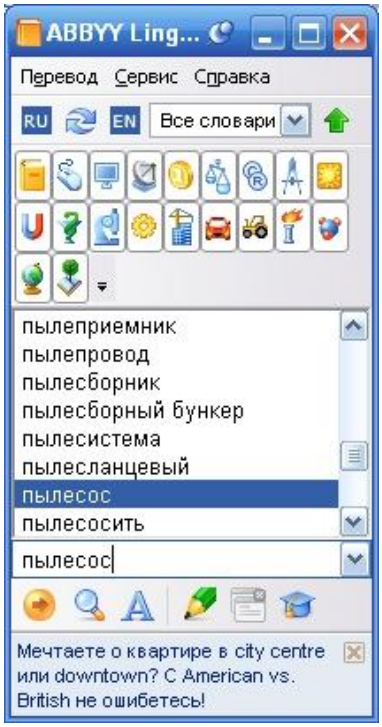


Рис. П5 ABBYY Lingvo. Переклад слова

3. Обчислення виразу$3 \times 23 - 56$.

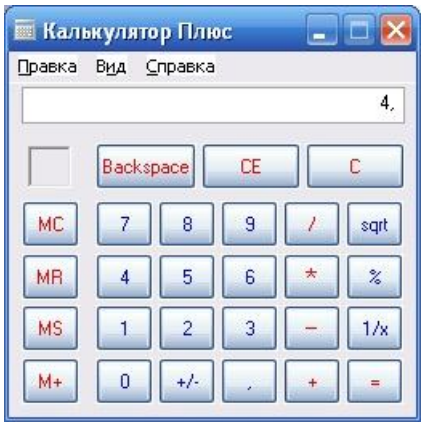


Рис. П6 Калькулятор Windows

4. Збереження файлу з новим ім'ям.

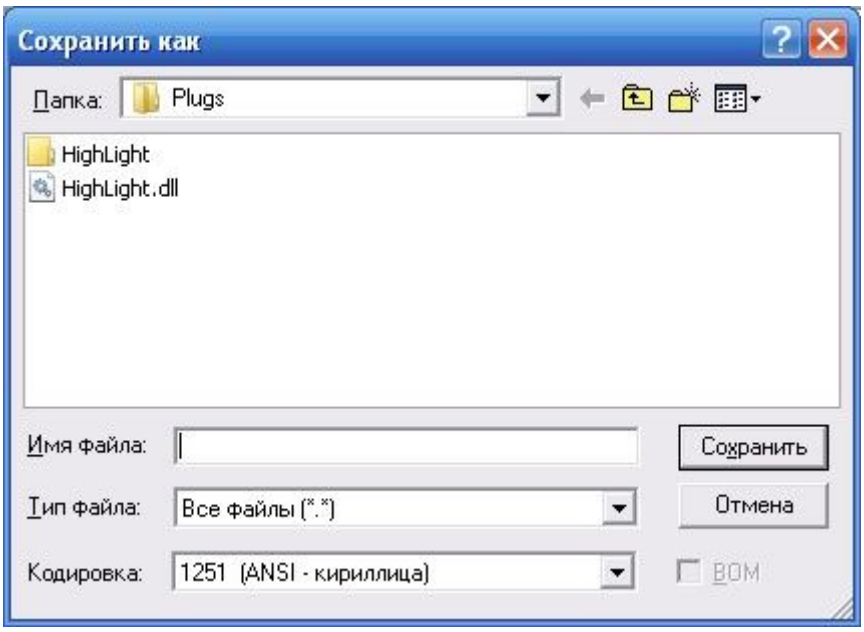


Рис. П7 Діалог збережень файлу Windows

Практичне завдання 9

Провести експрес-тест на визначення якості веб-навігації для сайтів обраних за бажанням.

Приклад виконання:

Сайт ogoloshennia.com.ua

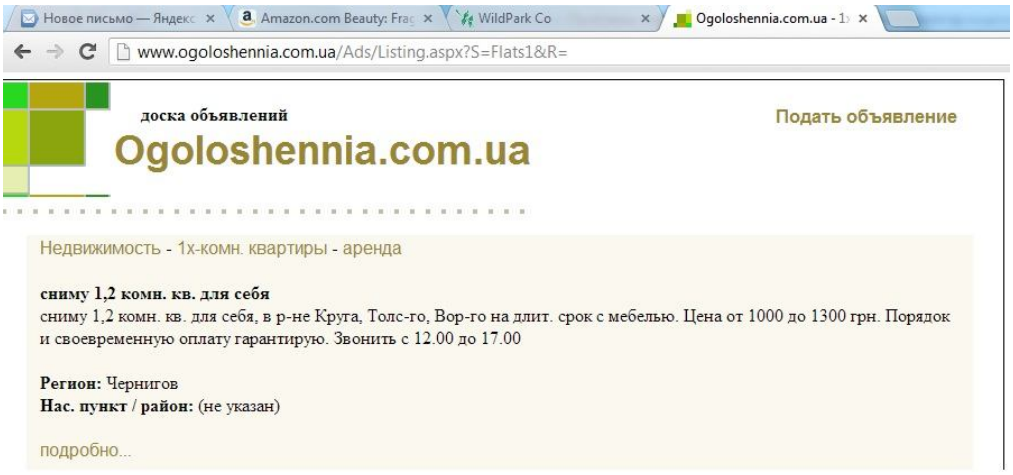


Рис. П8 Сайт, обраний для тестування якості веб-навігації

Веб-навігацію можна назвати хорошою, якщо дивлячись на будь-яку сторінку сайту можна відповісти на наступні питання:

1. Що це за сайт? (Логотип сайту);
2. На якій сторінці я перебуваю? (Назва сторінки);
3. Які головні розділи на цьому сайті? (Розділи);
4. Які опції є на цьому рівні? (Локальна навігація);
5. Де саме я перебуваю в загальній структурі сайту? (Покажчики «Ви знаходитесь тут»);
6. Яким чином здійснюється пошук?

Оцінка веб-навігації сайту ogoloshennia.com.ua.

1. Логотип сайту присутній. І сама назва сайту, і пояснювальний текст біля логотипу відразу дають зрозуміти про призначення сайту (+).

2. Назва сторінки відсутня. Підпис стрічки заголовку в браузері підібраний невдало, тому що назва сайту, яка і так є присутньою на логотипі, зайняла весь видимий простір. Про тему сторінки можна здогадатися тільки по «хлібних крихтах» (–).

3. Панель розділів відсутня. Але доступний основний сервіс даного сайту – «Подати оголошення» (+/–).

4. Локальної навігації немає (–).

5. Є вказівник місцерозташування типу «хлібні крихти» (+).

6. Пошук на даній сторінці недоступний (–).

Таким чином, з шести необхідних компонентів хорошої навігації на даній сторінці сайту присутні дві з половиною. Навігацію даного сайту не можна назвати задовільною.

Лабораторні роботи з рекомендаціями

Лабораторна робота № 1

Тема: ***Попереднє проектування користувацького інтерфейсу. Аналіз.***

Мета роботи:

- визначення цілей і задач користувачів;
- постановка завдання;
- пошук ідей для реалізації поставлених завдань.

Завдання до лабораторної роботи:

1. Вибрати й описати тему або сферу діяльності, роботу, проблему та ін. потенційних користувачів, яка буде аналізуватися. Визначити, на що буде зроблений натиск при проектуванні дизайну інтерфейсу: зміна існуючого способу здійснення діяльності користувачами (зміна); скорочення часу для розв'язку проблеми (час); спрощення аналізу й обробки необхідної користувачеві інформації (огляд).

2. Використовуючи метод спостереження провести аналіз розв'язку обраної проблеми трьома людьми, які мають різні соціальні характеристики. У звіті привести їхнє ім'я, вік, рід занять, основні риси характеру. Описати як вони вирішують проблему зараз, що вдається добре, які труднощі виникають, якими технічними й програмними засобами користуються для розв'язку даної проблеми, які пристрої й програмні засоби використовують взагалі. За бажанням представити фото спостережуваних, які ілюструють процес розв'язку проблеми.

3. На основі проведеного аналізу визначити й записати мінімум 15 цілей/потреб/задач користувачів.

4. Привести мінімум 5 прикладів існуючих дизайнів, ідеї яких можна використати для завдань, визначених у п.3.

Рекомендації з виконання роботи. *Теоретичні відомості знаходяться у частині 1, тема 2.*

Приклади тем для аналізу:

1. Довідник інформації про друзів з функцією нагадування (дні народження, дні весілля, професійні свята, інформація про дітей, чоловіка/жінку, батьків – їх імена, дні народження й т. п.).

2. Замітки про завдання побутового характеру на кожен день – що купити в магазині, що зробити у вихідний і т.д.

3. Менеджер фотографій для телефону (угруповання за темами, мінімальне редагування, розміщення фото на інтернет-ресурсах).

4. Система з вивчення йоги, гри на гітарі й т. п. (історія, пози, схеми, відео, програми для різних рівнів – все це організоване в одну систему).

5. Система керування файлами, які користувач скачує з Інтернету (угруповання за типами даних, за темами, мітки за типом: розібране, потрібне/непотрібне).

6. Домашній кабінет – угруповання часто використовуваних додатків і виклик їх з однієї програми гарячими клавішами або спеціальними кнопками (наприклад, з будь-якого місця фраза копіюється в домашній кабінет і далі на вибір користувача виконується або переклад, або пошук в Інтернеті, або пошук на певному сайті, або розрахунок і т.д.).

7. Електронні ноти – створення партитури, вибір партитури за автором, формою твору, ведення журналу останніх переглянутих партитур, додавання власних нотаток до партитури, автоматичне перегортання сторінки під час гри.

8. Підбір оптимального маршруту при подорожах з пересадженнями, на поїзді або літаку з врахування наступних вимог: мінімальний час очікування, мінімальна кількість пересаджень, оптимальна вартість.

9. Контроль строків придатності запасів продуктів у холодильнику (що з'їсти в першу чергу).

10. Система безоплатного обміну непотрібними речами.

11. Охорона праці при роботі з комп'ютером – додаток, який змушує робити перерви в роботі відповідно до нормативів.

12. Додаток для підрахунку суми покупки в супермаркеті в процесі додавання товарів у візок.

13. Організація заморозки продуктів на зиму. Наприклад, створення довідника за режимами заморозки або облік власних продуктів відправлених на заморозку: розробка режиму заморозки, умов зберігання, контроль строків придатності.

14. Додаток для створення звітів про відпустку – розміщення фото, угруповання фото за часом, географічним положенням, додавання коментарів, з можливість розміщення відповідних міток на карті.

Лабораторна робота № 2

Тема: ***Розробка користувальницьких сценаріїв. Прототипування.***

Мета роботи:

- одержання навичок швидкого прототипування;
- визначення життєвих ситуацій, у яких доведеться використовувати розроблювальне ПЗ;
- визначення технічних засобів, на яких ПЗ буде використовуватися;
- підбор варіантів користувацьких інтерфейсів для ПЗ.

Завдання до лабораторної роботи:

1. Описати вид діяльності, який аналізувався в ЛР №1, і яка дизайнерська задача буде вирішуватися в рамках цієї діяльності: зміна, час, огляд.

2. Описати дві користувацькі історії використання розроблювального додатку в 5-8 реченнях. Повинна бути чітко описана ситуація, задача, що розв'язується, показане її розв’язок за допомогою додатку й задоволення користувача. За бажанням історія може бути представлена у вигляді коміксу – по одній картинці на кожну пропозицію. Ситуації не мають бути подібними.

3. Сформувати остаточний список функціональних вимог. Навести користувацькі сценарії для кожної функції, враховуючи описані історії використання.

4. Створити два швидких прототипи користувальницького інтерфейсу додатку – нарисованих від руки або за допомогою додатку на сайті *balsamiq.com* або аналогічних. Не використовувати Photoshop й інші графічні редактори. Мета швидкого прототипування – показати навігаційну схему й розташувати основні функціональні елементи для кожної екранної форми.

Рекомендації щодо виконання лабораторної роботи. *Теоретичні відомості перебувають у частині 1, теми 2-5.*

Приклад користувацької історії у вигляді коміксу показаний на рис. Л2.1

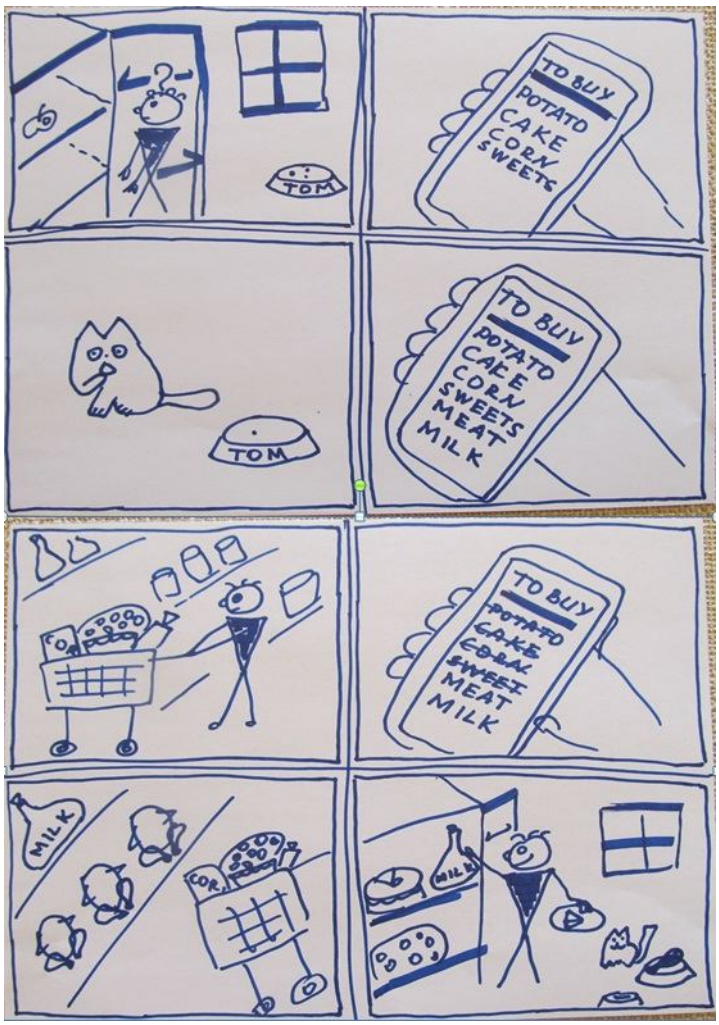


Рис. Л2.1 Історія використання ПЗ «Нотатки». Список покупок

Приклад фрагменту швидкого прототипу для ПЗ «Нотатки», виконаного за допомогою додатку Balsamiq показаний на рис. Л2.2. На екрані, який відкривається відразу після запуску додатку, показаний список всіх нотаток за 10 червня 2012 року. При виділенні нотатки біля неї з'являється кнопка Open (Відкрити), при натисканні на яку можна управляти вмістом нотатки – додавати/видаляти нові пункти, ставити мітки про виконання.

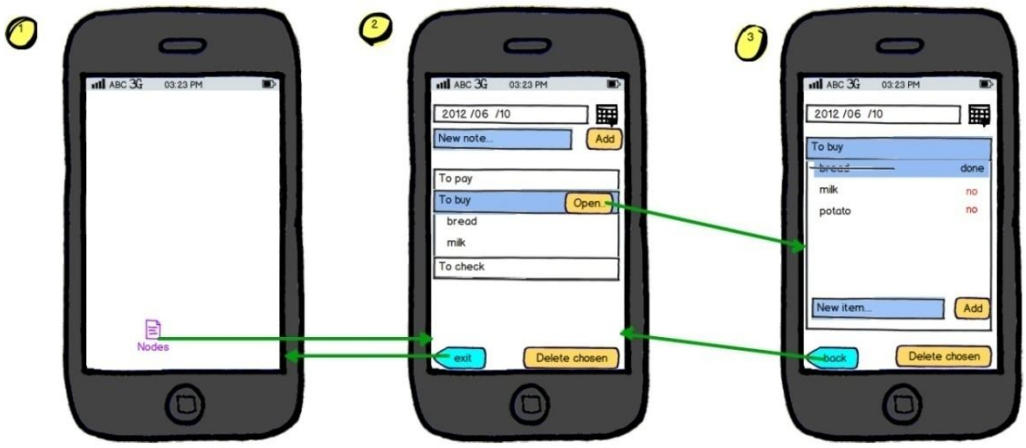


Рис. Л2.2 Прототип додатку «Нотатки» для смартфону.

Рис. Л2.3 Другий варіант прототипу додатка «Нотатки»

Другий прототип повинен містити ту ж функціональність, але пропонувати інший спосіб її реалізації. Так на рис. Л2.2 зміна стану про виконання здійснюється шляхом натискання на обраний пункт. На рис. Л2.3 представлений другий прототип, який реалізує функціональність роботи із вмістом нотатки. Але тут зміна стану про виконання відбувається шляхом перетаскування відповідного пункту в спеціальну область.

Лабораторна робота № 3

Тема: ***Кількісна оцінка інтерфейсу***

Мета роботи:

- одержання навичок кількісної оцінки інтерфейсу на прикладі моделі *GOMS;*
- оцінка ефективності й продуктивності розроблених прототипів інтерфейсу.

Завдання до лабораторної роботи:

Провести кількісну оцінку інтерфейсів прототипів ПО, розроблених у ЛР №2. Оцінити продуктивності й ефективність. Порівняти результати, отримані для кожного варіанта. Зробити висновки.

Рекомендації щодо виконання лабораторної роботи. *Теоретичні відомості перебувають у частині 1, тема 8 (табл. 4).*

Приклад розрахунку часу виконання сценарію.

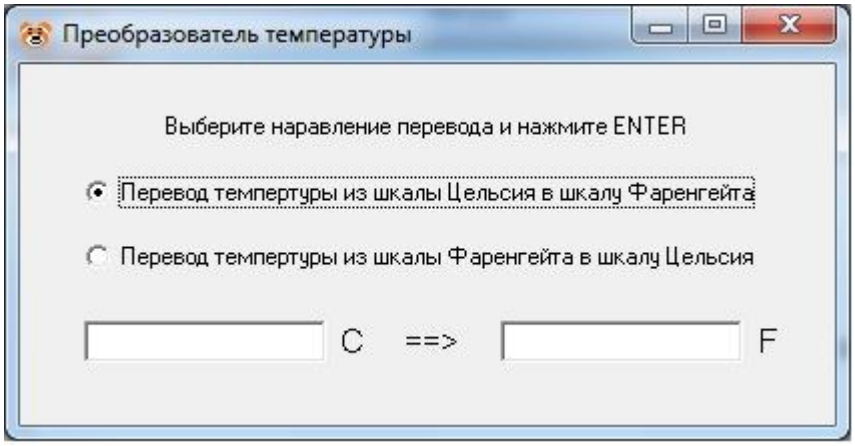


Рис. Л3.1 Вікно додатку «Перетворювач температури»

Вихідні дані: інтерфейс користувача, створений для задачі перетворення значення температури зі шкали за Цельсієм у шкалу за Фаренгейтом й навпаки (рис. Л3.1).

Для перекладу температури в даному варіанті необхідно виконати наступні операції:

- переміщення руки до ГПВ (*Н*);
- переміщення курсору до необхідного перемикача в групі (*НР*);
- натискання на необхідний перемикач (*НРК*);
- переміщення рук знову до клавіатури (*НРКН*);
- уведення чотирьох символів (*НРКНКККК*);

- натискання клавіші Enter (*НРКНКККККК*).

Після застосування правила 0 (табл. 4) для розміщення індексів *М* и одержано вираз:

НМРМКНМКМКМКМК.

Відповідно до правила 1 *РМК* замінюється на *РК* і відповідно до правила 2 видаляються оператори *М* усередині когнітивних одиниць. Вираз здобуває вигляд:

НМРКНМКККККМК.

Замінимо символи на тимчасові інтервали:

$K = 0{,}2$; $P = 1{,}1$; $H = 0{,}4$; $M = 1{,}35$.

Тепер додамо оператори й обчислимо сумарне значення часу:

$H + M + P + Д + H + M + K + K + K + K + M + K =$

$0{,}4 + 1{,}35 + 1{,}1 + 0{,}2 + 0{,}4 + 1{,}35 + (4 \times 0{,}2) + 1{,}35 + 0{,}2 = 7{,}15$ с.

Оцінка продуктивності елементів інтерфейсу на прикладі екранної форми з Лабораторної роботи № 2 (Рис. Л2.3). Як приклад розглянемо ситуацію, коли користувач хоче змінити стан нотатки. Є два стани: виконаний і невиконаний. У більшості випадків користувач буде міняти стан на виконаний. Назад можливо тільки у випадку помилки. Приймемо, що помилка можлива в 10% випадків. У такий спосіб зміна стану на «виконаний» має імовірність 0,9, на «невиконане» 0,1. Крім того необхідно вибрати пункт, стан якого треба змінити. В зв’язку з тим, що в даному варіанті прототипу пункти явно розмежовані за зонами, залежно від стану, то вибір варто розглядати в кожній області окремо. Всі пункти рівнозначні, тому вибір будемо вважати рівноймовірним. Для даної ситуації вибір пункту зі списку невиконаних 0,25, зі списку виконаних 0,5. Імовірності різних варіантів становлять:

– Зміна стану «невиконаного» пункту: $0{,}25 \times 0{,}9 = 0{,}225$;

– Зміна стану «виконаного» пункту: $0{,}5 \times 0{,}2 = 0{,}100$.

Інформаційний зміст розглянутого фрагмента інтерфейсу можна визначити за формулою:

$$0{,}225 \cdot \log_2\left(\frac{1}{0{,}225}\right) + 0{,}1 \cdot \log_2\left(\frac{1}{0{,}1}\right) = 2{,}14 + 7{,}67 = 9{,}81$$

Теоретично, якщо користувач вирішив змінити стан пункту, йому мінімально необхідно тільки визначити сам пункт. У цьому прикладі є всього 6 пунктів $\log_2 6 = 5{,}97$

Інформаційна продуктивність $E = \frac{5{,}97}{9{,}81} = 0{,}60$.

Лабораторна робота № 4

Тема: ***Розробка екранних форм користувацького інтерфейсу. Проектування візуального подання***

Мета роботи:

- вивчення процесу ітераційного проектування;
- створення діючого прототипу інтерфейсу;
- одержання навичок роботи із засобами розробки інтерфейсу.

Завдання до лабораторної роботи:

1. Вибрати один із прототипів, розроблених у лабораторній роботі № 2 як основний. Додати його до роботи. Якщо за основу вибирається комбінація двох прототипів, то додати до роботи обидва. Врахувати результати кількісної оцінки прототипів.

2. Написати не менш десяти пропозицій з вдосконалення прототипу. Обґрунтувати запропоновані зміни.

3. Створити діючу модель користувальницького інтерфейсу, яка забезпечує зворотний зв'язок з користувачем. Навести ілюстрації (скрін-шоти).

Рекомендації щодо виконання лабораторної роботи. *Теоретичні відомості перебувають у частині 1, теми 6, 7, 9.*

Лабораторна робота № 5

Тема: ***Юзабіліті тестування користувацького інтерфейсу***

Мета роботи:

- Вивчення методів юзабіліті-тестування;
- Одержання навичок організації й проведення юзабіліті-тестування;
- Аналіз результатів проведеного тестування.

Завдання до лабораторної роботи:

1. Навести скрін-шоти розробленого користувацького інтерфейсу.

2. Скласти план тестування, у якому визначити:

   а) що тестується (найменування й призначення ПО);

   б) що потрібно вивчити – проблеми, супересливі моменти, цілі, на яких потрібно зосередитися;

   в) час і місце проведення тестування, тривалість тестування;

   г) тестові завдання;

   д) методи одержання відклику від учасників тестування.

3. Скласти бланк завдань.

4. Скласти протокол проведення тестування, у якому надати:

   а) опис учасників (ПІБ, вік, рід занять) не менш 3-х чоловік;

   б) опис процесу тестування для кожної екранної форми кожним учасником окремо;

   в) час виконання завдання кожним учасником;

   г) фото процесу тестування додаються за бажанням.

5. Написати список змін (не менш десяти), які варто внести в інтерфейс за результатами тестування.

Рекомендації щодо виконання лабораторної роботи. *Теоретичні відомості перебувають у частині 1, тема 8.*

Приклад плану тестування ПЗ «Нотатки».

- що тестується:

Онлайн додаток «Нотатки» для мобільного телефону.

- що потрібно вивчити?

1. *Які функції додатку зрозумілі без пояснень.*
2. *Скільки часу потрібно для створення заповненої нотатки.*
3. *Скільки часу потрібно для пошуку певної нотатки.*

- місце проведення тестування:

Будь-яке кафе з безкоштовною wi-fi зоною.

- дата й тривалість тестування:

5.10.2013 з 18:00 до 19:00 (перше завдання)

6.10.2013 з 18:00 до 19:00 (друге завдання)

Загальна тривалість 2 години.

- завдання, які допоможуть вивчити поставлені проблеми:

1. Створити 20 листків нотаток. Заповнити два з них. В одному всі пункти відмітити як «виконані». Видалити 15 порожніх листків нотаток.

2. Створити 10 листків нотаток. Заповнити два з них. В будь-якій нотатці зі створених один пункт відзначити як «виконаний». Видалити цю нотатку. Видалити 15 нотаток.

- методи одержання відкликань від учасників тестування:

1. Міркування вголос;
2. Постінтерв'ю.

Приклад бланку завдань (для учасників тестування):

Даний додаток є електронною версією паперових листків для нотаток, які використовуються для швидкого створення списку короткочасних справ: кого привітати, що купити, за що заплатити й т. п. Цей додаток призначений для смартфону, щоб скористатися ним було можливо у будь-якім місці.

Завдання:

День перший.

Створити 20 листиків нотаток. Заповнити два з них. В одному всі пункти відмітити як «виконані». Видалити 15 порожніх листків нотаток.

День другий.

3. Створити 10 листків нотаток. Заповнити два з них. В будь-якій нотатці зі створених один пункт відзначити як «виконаний». Видалити цю нотатку. Видалити 15 нотаток.

Під час виконання завдань прохання коментувати всі свої дії вголос для кожного пункту завдання, відповідаючи на наступні питання:

Що ви бачите?

Як ви думаєте, для чого це?

Що ви збираєтеся робити?

Що вийшло в результаті?

Приклад фрагменту протоколу:

Учасники:

1. Олена. 30 років, еколог;
2. Віра. 35 років, методист вищої категорії;
3. Дмитро. 40 років, робітник;
4. Наталя. 30 років, власник туристичного агентства;

Тут приводиться тільки фрагмент частини протоколу з одержаного відгуку від учасників тестування (міркування вголос і інтерв'ю для учасника Олена).

Міркування вголос

Сторінка реєстрації

«Спочатку воно вимагало пароль. Це добре, значить про мої плани ніхто не довідається.»

Сторінка створення нотатки.

«Я бачу сторінку з текстовим полем і кнопкою Add, тобто Додати. Не відразу зрозуміла, що треба дати назву нотатці, без імені вона не додається. Нотатка не додається по натисканню клавіші Enter, тільки по натисканню на кнопку Add курсором миші. Це незручно, коли треба додати багато листків один за іншим. Після додавання з'являється створений листок з датою створення й кнопкою Del, мабуть Видалити. Якщо нотатка порожня, то поруч із назвою є мітка Empty. Так що я знаю які з них порожні. Якщо всі пункти в нотатці відзначені як «виконані», то поруч із назвою нотатки є мітка Завершене. Це добре. Нотатка видаляється одним натисканням на кнопку Del. Чим старше нотатка, тим нижче в списку вона перебуває.»

Сторінка заповнення нотатки

«Я додала завдання в нотатку. Біля неї з'явилася стрілочка вниз. Я вважаю це для того, щоб перемістити завдання у виконані. Нажала кнопку зі стрілочкою й завдання перемістилося у Виконані. Після додавання завдання я помітила, що єдиний спосіб його відредагувати це видалити й створити знову.»

Інтерв'ю

Питання:

Що відповідало очікуванням?

Що ні?

Коли були сумніви, що саме треба натиснути?

Які функції сподобалися?

Чого бракує?

Відповідь:

У цілому все було зрозуміло. Для мене було б більш зручно, якби нотатки розрізнялися за типами й мали різний пріоритет. А пункти можна було групувати, й вибирати різні кольори для завдань.

Результати за часом виконання завдань:

Завдання 1: 10 хвилин;

Завдання 2: 5 хвилин.

КОНТРОЛЬНІ ПИТАННЯ

1. Поняття інтерфейсу користувача.
2. Психологія користувачів.
3. Сприйняття й увага людини.
4. Властивості інтерфейсу користувача.
5. Вимоги до інтерфейсу користувача.
6. Базові принципи проектування, орієнтованого на користувача.
7. Людський фактор.
8. Основні принципи розробки інтерфейсу користувача.
9. Стандартизація інтерфейсу користувача.
10. Поняття життєвого циклу програмного продукту.
11. Типи діалогів.
12. Розробка сценарію діалогу.
13. Темп ведення діалогу.
14. Вибір візуальних атрибутів інформації, що відображується. Композиція й організація.
15. Вибір візуальних атрибутів інформації, яка відображується. Кольори.
16. Вибір візуальних атрибутів інформації, яка відображується. Шрифт.
17. Багатомірність екрану. Просторове розміщення візуальних елементів.
18. Використання звуку.
19. Використання анімації.
20. Профілі користувачів.
21. Типові проблеми інтерфейсу програмного забезпечення.
22. Методи попередження проблем інтерфейсу.
23. Властивості ефективного інтерфейсу.
24. Програмно-технічні засоби, які використовуються для реалізації користувальницького інтерфейсу.
25. Система показників оцінки ефективності людино-машинної взаємодії.
26. Місце й роль користувачів у системі.

27. Особливості графічного інтерфейсу.
28. Об'єктний підхід для проектування інтерфейсу.
29. Компоненти графічного інтерфейсу.
30. Взаємодія користувача з додатком.
31. Проектування піктограм.
32. Структура первинного вікна, основні операції з вікнами.
33. Багатодокументний інтерфейс.
34. Вибір моделі вікна.
35. Основні властивості вторинних вікон.
36. Панелі властивостей і контроль параметрів.
37. Діалогові панелі.
38. Головне меню й меню, що випадає.
39. Спливаючі й каскадні меню.
40. Заголовок меню. Пункти меню.
41. Кнопки управління, перемикачі, прапорці.
42. Список одиничного вибору, список, що випадає.
43. Розширений список і список множинного вибору.
44. Текстові поля, багаторядкові текстові поля.
45. Комбінований список.
46. Дискретне текстове поле. Статичні текстові області.
47. Поле призначення гарячих клавіш.
48. Панель інструментів і рядок станів.
49. Смуги прокручування, повзунковий регулятор.
50. Індикатор стану процесу, область повідомлень.
51. Контекстна допомога.
52. Проблемно-орієнтована допомога.
53. Довідник.
54. Майстер.
55. Засоби адаптації інтерфейсу користувача.
56. Мультимедіа як частина інтерфейсу користувача.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. **Акчурин, Э. А.** Человеко-машинное взаимодействие [Текст] / Э. А. Акчурин. – М.: Солон-пресс, 2008 – 93 с.
2. **Гультяев, А. К.** Проектирование и дизайн пользовательского интерфейса [Текст] / А. К. Гультяев, В. А. Машин. – СПб, КОРОНА принт, 2000. – 349 с.
3. **Иттен**, И. Искусство цвета [Текст] / Иттен Иоханнес – Пер. с немецкого М.: Изд. Д.Аронов, 2001. – 96 с.
4. **Круг, С.** Веб-дизайн: книга Стива Круга или «не заставляйте меня думать!» [Текст] / Стив Круг. – Пер. с англ. – СПб: Символ Плюс, 2008. – 224 с.
5. **Купер, А.** Психбольница в руках пациентов. [Текст] / Алан Купер – Пер. с англ. М.: Символ-Плюс, 2009. – 336 с.
6. **Мандел, Т.** Разработка пользовательского интерфейса [Текст] / Тео Мандел – Пер. с англ. – М.: ДМК Пресс, 2001. – 146 с.
7. **Мацяшек, Л. А.** Анализ требований и проектирование систем. Разработка информационных систем с использованием UML [Текст] / Лешек А. Мацяшек – Пер. с англ. М.: Вильямс, 2002. – 432 с.
8. **Раскин, Д.** Интерфейс: новые направления в проектировании компьютерных систем [Текст] / Джеф Раскин – Пер. с англ. М.: Символ-Плюс, 2009. – 272 с.
9. **Сергеев, С. Ф.** Введение в проектирование интеллектуальных интерфейсов [Текст] : учебное пособие./ С. Ф. Сергеев, П. И. Падерно, Н. А. Назаренко – СПб: СПбГУ ИТМО, 2011. – 108 с.
10. **Соммервил, Й.** Инженерия програмного обеспечения. [Текст] / Йан Соммервил : Пер. с англ. - М.: Издательский дом «Вильямс», 2002 – 624 с.
11. **Торрес, Р. Дж.** Практическое руководство по проектированию и разработке пользовательского интерфейса [Текст] / Р. Дж. Торрес – Пер. с англ. М.: Вильямс, 2002. – 401 с.

12. **Уэйншенк, С.** 100 главных принципов дизайна [Текст] / Сьюзан Уэйншенк – Пер. с англ. СПб.: Питер, 2012. – 272 с.
13. **Cooper, A.** About Face 3: The Essentials of Interaction Design [Text] / Alan Cooper, Robert Reimann, and David Cronin. – Wiley Publishing, Inc, 2007 – 651 pp.
14. **Crumlish, C.** Designing Social Interfaces [Text] / Christian Crumlish, Erin Malone – Gravenstein Highway North, Sebastopol: O'Reilly Media, Inc, 2009 – 518 pp.
15. **Nielsen, J.** Designing Web Usability [Text] / Jacob Nielsen – Indianapolis, IN: New RidersPublishing, 2000. – 419 pp.
16. **Scott, B.** Designing Web Interfaces [Text] / Bill Scott, Theresa Neil – Gravenstein Highway North, Sebastopol: O'Reilly Media, Inc, 2009 – 334 pp.
17. **Tidwell, J.** Designing Interfaces [Text] / Jenifer Tidwell – Gravenstein Highway North, Sebastopol: O'Reilly Media, Inc, 2010, – 577 pp.