Бугаєць Н.О.
НПУ імені М.П. Драгоманова

Програмування візуально-орієнтованого інтерфейсу в програмі Maple

Інформатизація як вирішальний фактор розвитку сучасного суспільства вимагає підготовки кваліфікованих кадрів, здатних використовувати в своїй професійній діяльності всі досягнення в галузі інформаційних технологій.

Серед сучасних інформаційних технологій особливе місце займають технології програмування. Візуальне програмування – один з сучасних напрямів програмування. В його основі лежить об'єктно-орієнтований підхід до описання процесів (явищ), який є одним з найбільш ефективних та зручних і використовується сьогодні програмістами для створення великих програмних систем.

Вміння працювати в середовищі візуального програмування є перспективним компонентом в системі професійних компетентностей студентів фізико-математичних спеціальностей. Володіння навичками візуального програмування підвищує професійний рівень використання сучасних прикладних програм, дозволяє конструювати для них додатки, які застосовуються в практичній сфері діяльності.

Для візуально-орієнтованого програмування важливими є поняття «інтерфейс користувача», «графічний інтерфейс користувача».

В тлумачних словниках [1], [2] *інтерфейс користувача (user interfase)* означено як спосіб управління комп'ютерною системою з боку користувача.

В [3] інтерфейс користувача розглядається як сукупність інформаційної моделі проблемної галузі, засобів і способів опрацювання користувачем такої інформаційної моделі, а також компонентів для забезпечення формування інформаційної моделі в процесі роботи з програмною системою.

У 80-х р.р. XX ст. інтерфейси користувача були текстовими або створювалися у вигляді спеціальних форм. На даний час практично на всіх комп'ютерах підтримується графічний (візуально-орієнтований) інтерфейс користувача.

Графічний користувацький інтерфейс (Graphical User Interface, GUI) – інтерфейс, в основі якого лежить візуалізація об'єктів, з якими працює користувач, а також самого процесу опрацювання таких об'єктів [2].

Грамотно спроектований інтерфейс користувача вкрай важливий для успішної роботи з програмним засобом. Тому розробник програмного додатку повинен знати основні принципи проектування ефективного інтерфейсу користувача. Ефективність програми полягає у її реальній віддачі за умов вмілого користування нею і визначається відношенням позитивного результату її використання до затрачених зусиль.

Процес проектування інтерфейсу повинен орієнтуватися на користувача і здійснюватися за такими принципами [3]:

1. Природність (інтуїтивність) інтерфейсу. Природний інтерфейс означає, що користувач не повинен істотно змінювати звичні для нього способи розв'язування задачі. Це, зокрема, означає, щодо повідомлень та результатів, які з'являються при роботі з програмним додатком, не повинні вимагатися додаткові пояснення. Доцільно також зберегти систему позначень та термінологію, які використовуються в даній предметній галузі.
2. Узгодженість інтерфейсу – дозволяє користувачам застосовувати наявні знання для розв'язування нових завдань та зосереджувати увагу на розв'язуваній задачі, а не витрачати час на вияснення відмінностей у використанні тих чи інших елементів управління, команд тощо. Узгодженість робить інтерфейс пізнаваним та передбачуваним.
3. «Люб'язність» інтерфейсу. Навіть при наявності добре спроектованого інтерфейсу користувачі можуть допускати ті чи інші помилки. На кожному етапі роботи повинен дозволятися тільки певний набір дій і видаватися попередження користувачів про ситуації, де можуть пошкодитися дані. Ще краще, якщо у користувача є можливість відмінити або виправити виконані дії. Ефективний інтерфейс повинен забезпечувати можливість користувачеві попередити ситуації, які ймовірно закінчаться з помилками. В інтерфейс можуть бути вбудовані засоби підтримки користувача, які забезпечать різні рівні допомоги та довідки.
4. Принцип зворотного зв'язку. Кожна дія користувача повинна отримувати візуальне, а іноді звукове підтвердження того, що програмний засіб «зреагував» на введену команду.
5. Простота інтерфейсу. Мається на увазі не спрощеність, а забезпечення легкості у вивченні та використанні інтерфейсу. Крім того повинен без перешкод надаватися доступ до всього переліку функціональних призначень, передбачених у даному програмному додатку. Один з

важливих шляхів підтримки простоти – лаконічне подання на екрані повідомлень, необхідних для виконання користувачем певного кроку завдання. Багатослівні командні імена або повідомлення, непродумані або непотрібні фрази можуть ускладнювати отримання важливих даних. Інший шлях до створення простого, але ефективного інтерфейсу – розміщення та подання на екрані елементів, з урахуванням їх змістового значення та логічних взаємозв'язків.

6. Гнучкість інтерфейсу. Полягає в здатності враховувати рівень підготовки та результативність праці користувача. Гнучкість передбачає можливість зміни системи управління виконанням завдань чи вхідних даних.
7. Естетична привабливість. Потрібно щоб середовище, яке формується на екрані, не тільки сприяло розумінню користувачем поданих повідомлень і даних, але і дозволяло б зосередитися на найважливіших особливостях даних.

Крім загальних принципів проектування інтерфейсу в кожній галузі є свої спеціальні особливості, яких слід дотримуватися при розробці програмних додатків.

Наприклад, проектування навчальної програми повинно базуватися на певному психолого-педагогічному фундаменті. Передусім необхідно спроектувати процес навчання і лише потім здійснювати його комп'ютерну реалізацію [5].

До навчальних програм ставляться педагогічні вимоги: дидактичні, методичні, обґрунтування вибору тематики, перевірка на педагогічну доцільність використання, врахування вікових та індивідуальних особливостей учнів.

М.І. Жалдак зазначає, що сучасний розвиток програмного забезпечення комп'ютерів досяг такого рівня, коли в багатьох випадках алгоритм досягнення мети може бути побудований автоматично. При цьому вказівки комп'ютерові потрібно задавати в термінах шуканих результатів, а не в описах процесів, що приводять до таких результатів. Головна трудність полягає в тому, щоб кваліфіковано і точно охарактеризувати шукані результати, що висуває відповідні вимоги до загальної строгості і логічності мислення користувача. Особливого значення при використанні інформаційно-комунікаційних технологій в навчальному процесі набуває врахування і розвиток неформалізовних, творчих компонентів мислення: реалізація проблемної ситуації чи постановка задачі; самостійне вироблення критеріїв добору потрібних операцій, що приводять до розв'язку та ін. [4].

Програма Maple – одна з небагатьох систем комп'ютерної математики, яка включає засоби програмування інтерфейсу, які можна використовувати для розробки програмних додатків навчального призначення з графічним інтерфейсом.

В останніх версіях програми (починаючи з Maple 8) було введено новий пакет Maplets, призначення якого – програмування візуально-орієнтованого (графічного інтерфейсу) для виконання обчислень із *застосуванням процедур Maple*, запитів та повідомлень. Такі додатки прийнято називати маплетами (maplets, Maplet Application). Якщо в процедурному програмуванні головною метою було отримати результат і вивести його на екран у звичайному вигляді, то при роботі з маплетами оформлення результатів на екрані відіграє важливу роль. Результати розміщуються у характерному для операційної системи Windows вікні, де можна застосувати різноманітні елементи управління, властиві для вікон прикладних програм: текстові поля, поля редагування, поля-списки, кнопки, слайдери тощо.

Пакет Maplets створений на основі застосування засобів мови Java. Але для створення маплетів не обов'язково нею володіти, достатньо знати основи програмування в Maple та вміти використовувати засоби пакету Maplets.

Пакет Maplets містить одну функцію Display, яка необхідна для показу і управління створеним маплетом, та підпакети:

1) Elements (Елементи) – містить команди для створення та налаштування окремих елементів маплету, наприклад, вікон, кнопок, текстових та графічних полів, списків, слайдерів, таблиць, меню, розмітки вікна тощо;
2) Examples (Приклади) – команди використання заготовок, зокрема, вікна з попередженням, вказівкою чи запитом, вікна налаштування кольору, вибору шляху до файлу та ін.;
3) Tools (Інструменти) – допоміжні засоби для розробки маплетів;
4) Utilities (Послуги) – сервісні команди виведення вікон з повідомленням про помилки, відкривання файлу, довідки.

В таблиці 1 подано основні функції програмування візуально-орієнтованого інтерфейсу, які входять до даних підпакетів.

Таблиця 1

Пакет	***Функції***

Elements	Windows Body Elements (Елементи вікна)	Button, RadioButton, CheckBox, RadioBatton, Slider, TextField, TextBox, ComboBox, Label, Plotter, Table, MathMLEditor, MathMLViewer, ToggleButton, DropDownBox
	Layout Elements (Елементи розмітки)	BoxCell, BoxColumn, BoxLayout, BoxRow, GridCell, GridRow, GridLayout, HorizontalGlue, VerticalGlue, BorderLayout
	Menu Elements (Елементи меню)	CheckBoxMenuItem, RadioButtonMenuItem, PopupMenu, Menu, MenuBar, MenuItem, MenuSeparator
	Command Elements (Командні елементи)	CloseWindow, Evaluate, RunDialog, RunWindow, SetOption, Shutdown
	Toolbar Elements (Елементи-інструменти)	ToolBar, ToolBarButton, ToolBarSeparator
	Dialog Elements (Елементи управління«діалогу»)	AlertDialog, ColorDialog, ConfirmDialog, FileDialog, InputDialog, MessageDialog, QuestionDialog
	Інші елементи	Action, Argument, ButtonGroup, Font, Image, Item, MapletReturn, ReturnItem, TableHeader, TableItem, TableRow,Window
Examples	Alert, Confirm, GetColor, GetEquation, GetExpression, GetFile, GetInput, Integration, KernelOpts, Message, Question, Selection, ShowTable, SignQuery	
Tools	AddAttribute, AddContent Get, ListBoxSplit, Print, Set, SetTimeout, StartEngine, StopEngine	
Utilites	ErrorDialog, GetFile, HelpBrowser	
Display		

Розглянемо на прикладі, що таке маплет і як він створюється.

Завантажуємо пакет Elements:

```
restart;
with(Maplets[Elements]):
```

Далі пишемо нескладну програму.

maplet_diff:=Maplet(Window('title'="Диференціювання", # створюємо вікно з назвою ”Диференціювання”;

["Введіть f(x):", TextField['TFfx']()], # створюємо текстове поле для введення функції, яку диференціюємо;

["Змінна диференціювання:", TextField['TFx'](3)], # створюємо текстове поле розміром 3 одиниці для введення змінної диференціювання;

TextField['TFf1x']('editable'='false', 3..40), # створюємо поле для виведення результату;

[Button("Диференціювати",Evaluate('TFf1x'= 'diff(TFfx,TFx)')), # кнопка для виконання операції диференціювання;

Button("Ok", Shutdown(['TFfx', 'TFx', 'TFf1x']))], # кнопка для закривання вікна;

Button("Очистити", SetOption('TFfx'=" ")] # кнопка для підготовки вікна до нової операції диференціювання;

Maplets[Display](maplet_diff): # демонстрація та виконання маплету.

В результаті виконання описаного маплету отримаємо вікно, показане на рис. 1.

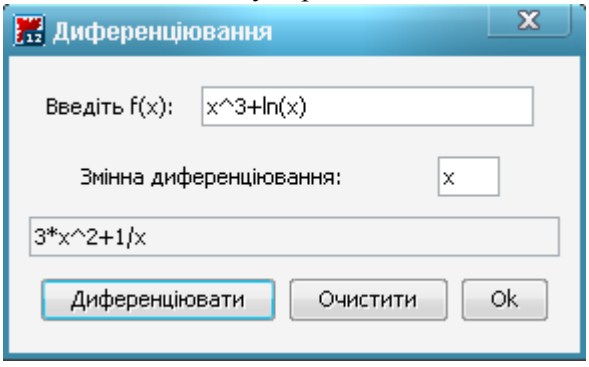

Рис. 1

Маплет можна зберегти як окремий програмний додаток, який відкривається за допомогою MapletisViewer і не потребує запуску програми Maple. Для цього треба виконати послугу *File/Export As…* та зберегти як файл з розширенням **.maplet*.

Таким чином, використовуючи процедури пакету Maplets, можна створювати додатки з елементами візуально-орієнтованого інтерфейсу, що допоможе користувачеві виконувати обчислення, навіть не знаючи тонкощів роботи з Maple.

Маплети можуть використовуватися як зручний засіб автоматизації опрацювання експериментальних даних в процесі проведення математичних досліджень. Їх використання суттєво полегшує налаштування та опис змінних і параметрів в процедурах Maple.

Розглянемо маплет «Методи обчислення інтегралів», який створено з метою підтримки навчання методів чисельного інтегрування (рис. 2). В поле $f(x)$: вводимо вираз функції, в поля a: і b: – межі інтегрування. На слайдері h: вибираємо крок поділу інтервала $[a;b]$. Натискаючи на відповідні кнопки, отримуємо графічне зображення результатів інтегрування за тим чи іншим методом.

Використовуючи даний маплет, легко скласти таблицю значень інтеграла функції, обчисленого за квадратурними формулами, трапецій, Сімпсона, за методами, використання в Maple (таблиця 2). Порівнюючи отримані дані, можна дослідити, як змінюються значення інтегралу при зменшенні кроку h та подрібненні інтервалу інтегрування, які формули дають результат найвищої точності, чим це зумовлено.

Таблиця 2

$\int_0^1 e^x dx$	**h**	**Середніх**	**Трапецій**	**Сімпсона**	**Maple**
	10	1.717566086	1.719713491	1.718282782	1.718281828
	20	1.718102854	1.718639789	1.718281888	1.718281828
	30	1.718202281	1.718440926	1.718281841	1.718281828
	40	1.718237082	1.718371321	1.718281833	1.718281828

Таким чином, навички роботи із засобами візуально-орієнтованого програмування інтерфейсу в Maple дають можливість викладачам та вчителям самим розробляти навчальні, демонстраційні, контролюючі та інші педагогічні програмні засоби досить високого дидактичного рівня. Причому на основі зауважень, які виникають в процесі випробування маплетів навчального призначення, можна оперативно внести зміни для вдосконалення програмного додатку, налаштувати його під різні умови використання.

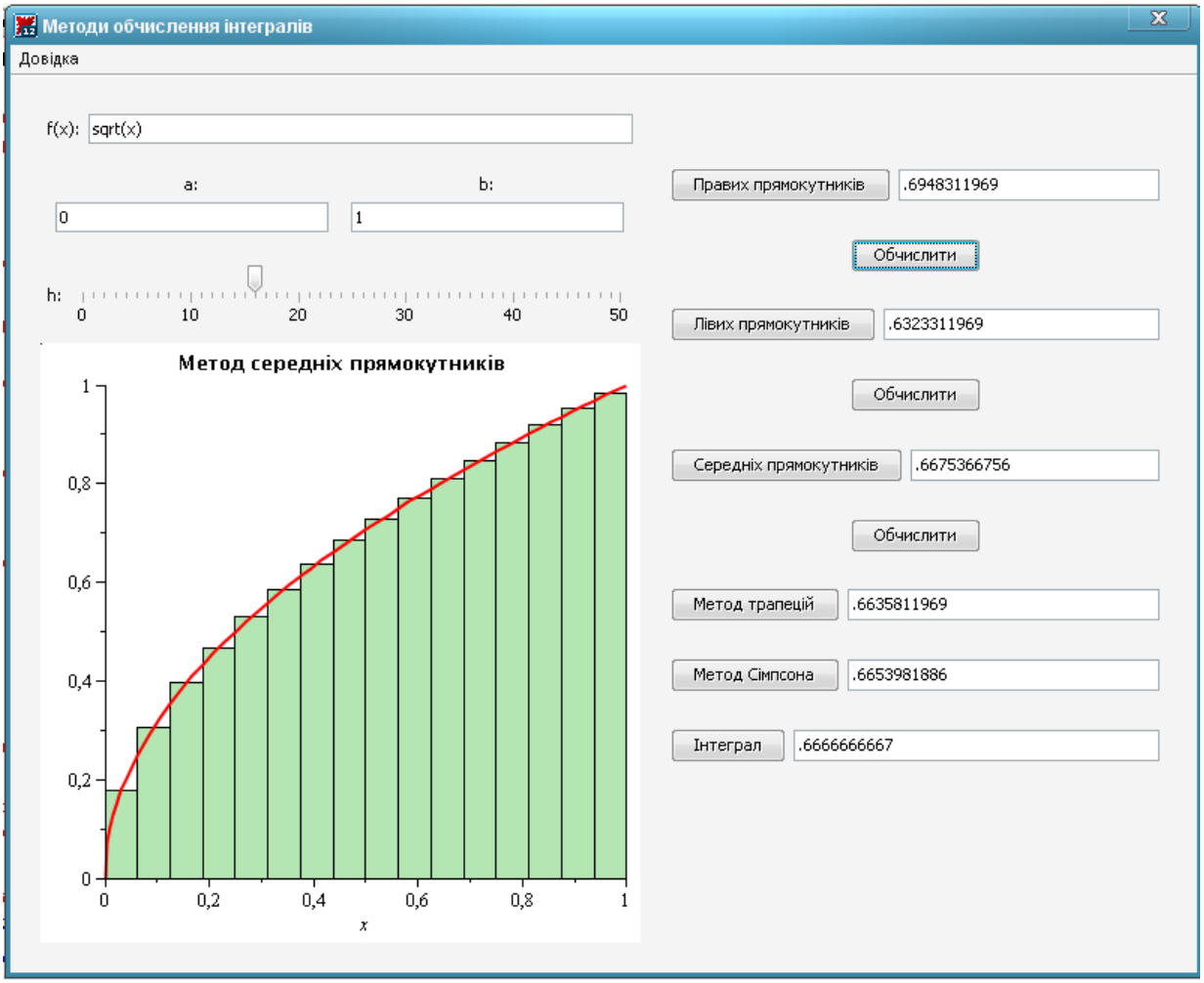

Рис. 2

З іншого боку, розробляючи свої оригінальні маплети, студенти не тільки знайомляться з принципами та технологіями візуального програмування, а також мають можливість найбільш повно розкрити свій творчий та інтелектуальний потенціал.

Література

1. Англо-український тлумачний словник з обчислювальної техніки, інтернету і програмування / Е.М. Пройдаков, Л.А. Теплицький. – К.: Софт Прес, 2006. – 549 с.

2. Великий тлумачний словник української мови: 170000 слів і словосполучень / Уклад. і голов. ред. В.Т. Бусел. – К.: Ірпінь: ВТФ «Перун», 2004. – 1440 с.

3. Гультяев А., Машин В. Проектирование и дизайн пользовательского интерфейса. – СПб.: Корона принт, 2000. – 352 с.

4. Жалдак М.І., Горошко Ю.В., Вінниченко Є.Ф. Математика з комп'ютером. Посібник для вчителів. – К: НПУ ім. М.П. Драгоманова, 2009. – 282 с.

5. Машбиц Е.И. Психолого-педагогические проблемы компьютеризации обучения. – М.: Педагогика, 1988. – 192 с.

6. M.B. Monagan, K.O. Geddes, K.M. Heal, G. Labahn, S.M. Vorkoetter, J. McCarron, P. DeMarco. Maple Introductory Programming Guide. – Maplesoft, a division of Waterloo Maple Inc, 2008. – 381 p.