МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВОЛОДИМИРА ДАЛЯ
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ТА ЕЛЕКТРОНІКИ
КАФЕДРА ПРОГРАМУВАННЯ ТА МАТЕМАТИКИ

Пояснювальна записка
до дипломної роботи
_____бакалавр _____
(освітньо-кваліфікаційний рівень)

на тему «Розробка інтерактивної програми контролю знань шкільних дисциплін»

Виконав: студент 4 курсу, групи ІТ-651
напряму підготовки 6.050103 «Програмна інженерія»

_____ Холоденко В.В
(підпис)

Керівник,
професор, д.т.н._____ Марченко Д.М.
(підпис)

Рецензент,
доцент, к.т.н._____ Митрохін С.О.
(підпис)

СЄВЄРОДОНЕЦЬК – 2019

СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

ІМЕНІ ВОЛОДИМИРА ДАЛЯ

Факультет інформаційних технологій та електроніки
Кафедра програмування та математики
Освітньо-кваліфікаційний рівень бакалавр
Напрям підготовки 6.050103 «Програмна інженерія»

ЗАТВЕРДЖУЮ
Завідувач кафедри ПМ,
д.т.н., доцент
_____ Лифар В.О.
«___» _____ 2019 р.

З А В Д А Н Н Я
НА ДИПЛОМНУ РОБОТУ СТУДЕНТУ
Холоденка В.В

1. Тема роботи Розробка інтерактивної програми контролю знань шкільних дисциплін.
керівник роботи : проф., д.т.н. Марченко Д.М.
затверджені наказом вищого навчального закладу від 23.04.2019 р. №68/14.04
2. Строк подання студентом роботи 07.06.2019 р.
3. Вихідні дані до роботи

Об'єктом даної роботи є навчальний процес середніх шкіл.

3.1 Літературні джерела:

Горнаков, С.Г. Инструментальные средства программирования и отладки шейдеров в DirectX - СПб. : БХВ-Петербург, 2005

Маккинли,Уэс. Python и анализ даннях - Москва : ДМК Пресс, 2015.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

4.1 Вступ

4.2 Аналіз предметної галузі (огляд літератури), з висвітленням наступних питань:

Історія розвитку комп’ютерних ігор

Що таке DirectX?

Мова Python

4.3 Основна частина, в якої висвітлити:

Сутність поняття «контроль»

Використання ЕОМ для контролю

Використати Construct classic для розробки програми

4.4 Висновки

4.4 Перелік використаних джерел

5. Перелік графічного матеріалу немає

6. Дата видачі завдання 23.04.2019 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломної роботи	Строк виконання етапів роботи	Примітка
1	Одержання завдання на виконання роботи	23.04.19	
2	Укладання і погодження з керівником плану і етапів виконання роботи	23.04.19	
3	Узагальнення даних літературних джерел, укладання розділу «Аналіз предметної галузі»	25.04.19	
4	Аналіз шляхів виконання завдання. Вибір і погодження з керівником оптимального шляху	09.05.19	
5	Проектування інфологічної моделі задачі що реалізується.	20.05.19	
6	Укладання та тестування програмного продукту	30.05.19	
7	Укладання, оформлення та погодження пояснювальної записки з керівником	02.06.19	
9	Здача готової пояснювальної записки на кафедру	10.06.19	
10	Укладання доповіді і презентації	11.06.19	

Студент _____ Хололоденко В.В.
(підпис)

Керівник роботи _____ Марченко Д.М.
(підпис)

ЛИСТ ПОГОДЖЕННЯ І ОЦІНЮВАННЯ

бакалаврської роботи студента гр. ІТ-651 Холоденка В.В.

Науковий керівник
Професор, д.т.н._____Марченко Д.М.

Оцінка наукового керівника:_____

Рецензент Митрохін С.О., доцент каф. ПМ СНУ ім. В. Даля
ПІБ, місто роботи, посада

Оцінка рецензента:_____

Кінцева оцінка за результатами захисту: _____

_____

Голова ЕК,
Зав. кафедри ПМ
д.т.н., доцент _____Лифар В.О.
підпис

РЕФЕРАТ

Текст – 63 с., рис. – 10, табл. – 2, додатків – 0, літературних джерел – 22

Мета роботи - є розгляд безпосередньо складових поняття «контроль», а також встановлення доцільність використання комп’ютера для здійснення контролю знань.

Проаналізувано основні функції та призначення контролю в навчанні, визначино сутність і особливості розробки тестового контролю.

Ключові слова: КОНТРОЛЬ, СУТНІСТЬ, ЗВ'ЯЗОК, ТЕРМІН, АКТОР, ПІШАК, СВІТ, DIRECT3D, РЕНДЕРІНГ, ТЕССЕЛЯЦІЯ, ОЦІНКА, CONSTRUCT CLASSIK.

ЗМІСТ

ВСТУП

Контроль знань учнів є складовою частиною процесу навчання. За означенням контроль – це відношення досягнутих результатів із запланованими цілями навчання. Деякі вчителі традиційно підходять до організації контролю, використовуючи його в основному заради показників досягнутого. Перевірка знань учнів повинна давати відомості не лише про правильність чи неправильність кінцевого результату виконаної діяльності, але і про неї саму: чи відповідає форма дій даному етапу навчання.

В наш час, коли інтенсивно використовуються інформаційні технології в повсякденному житті людини, в сфері освіти також впроваджується експлуатація електронно-обчислювальної машини як для наочності навчального матеріалу, так і для контролю знань, умінь і навичок. В основному контроль знань проводиться в режимі тесту. Використання ЕОМ для опрацювання результатів контролю знань потребує одержання числової відповіді в задачі. Це скорочує можливі помилки операторів при введенні цих результатів у пам’ять ЕОМ.

За 120 років існування тестології накопичено значний досвід використання тестів у різноманітних сферах людської діяльності, в тому числі й освіті. В Україні цей спосіб оцінювання став актуальним завдяки його широким можливостям: здатності охоплювати значний обсяг навчального матеріалу, об’єктивності процедури оцінювання, врахуванні індивідуальних особливостей учнів, уведенню незалежного оцінювання як форми підсумкового контролю якості підготовки випускників. З огляду на це, створення інструментарію для виявлення знань і вмінь учнів шляхом тестування є актуальним. Комп'ютерне тестування успішності надає можливість реалізувати основні дидактичні принципи контролю навчання: індивідуального характеру перевірки й оцінки знань; системності перевірки й оцінки знань; тематичності; диференційованої оцінки успішності навчання; однаковості вимог; принцип об'єктивності.

Актуальність обраної нами теми визначається недосконалістю контролю в сучасній освіті шкіл.

Протиріччям в роботі є можливість і в деяких випадках необхідність використання тестового контролю для отримання найбільш докладної і повноцінної інформації про рівень знань, умінь і навичок.

Все це визначило тему нашої дипломної роботи: «Розробка інтерактивної програми контролю знань шкільних дисциплін».

Об’єкт досліджень: тестовий контроль в сучасній освіті.

Предмет досліджень: сутність контролю, мета і його функції.

.Мета роботи: розглянути безпосередньо складові поняття «контроль», а також встановити доцільність використання комп’ютера для здійснення контролю знань.

РОЗДІЛ 1. АНАЛІТИЧНИЙ ОГЛЯД

1.1 Історія розвитку комп'ютерних ігор

Фактично першою комп'ютерної, або відео грою можна вважати Rocket Simulator або Ракетний Симулятор. Ракетний симулятор - розважальний апарат на базі електронно-променевої трубки, що імітує управління польотом ракети. Томас Т. Голдсміт-мл. і Естл Рей Манн розробили пристрій в 1947 році.

Ракетний симулятор, який вважається найбільш ранній з відомих інтерактивних електронних ігор, з'явився передвісником комп'ютерних ігор. Однак сам апарат не мав цифрового процесора для обробки інформації, а використовував аналогові ланцюги для управління електронно-променевою трубкою і формування зображення на екрані. По виду він нагадував радар часів Другої світової війни, а для прицілювання використовувалися екранні накладання.

25 січня 1947 року автори винаходу подали заявку на патент, який був ними отриманий 14 грудня 1948 року. Однак в масове виробництво і продаж апарат так і не надійшов.

Наступне важлива подія відбулася в 1952 році - була створена перша комп'ютерна гра, яка виводилася на растровий дисплей і оброблялася в обчислювачі, а не за допомогою аналогових ланцюгів. Гру назвали OXO (хрестики-нулики) - комп'ютерна гра для комп'ютера EDSAC, що представляє собою хрестики-нулики. Розроблено в 1952 році А. С. Дугласом як ілюстрація до кандидатської дисертації на тему взаємодії людини і комп'ютера.

У OXO людина грав проти комп'ютера, виставляючи хрестик або нулик в потрібну клітину поля за допомогою дискового номеронабирача. Висновок здійснювався на растровий дисплей розмірністю 35 × 16 точок. Символ і черговість ходу вибиралися гравцем до початку гри.

OXO не отримала широкого поширення, так як EDSAC був унікальним комп'ютером, що знаходяться в бібліотеці Кембріджського університету.

Першим мережевим шутером є Empíre (рус. Імперія) - комп'ютерна гра для системи PLATO, створена в 1973 році Джоном Далеске. Гра з великою часткою ймовірності є першою в жанрі мережевого багатокористувацького шутера, а також одним з перших мережевих екшенів. Хоча термінали PLATO мали тач-панелі для введення даних, у них був відсутній маніпулятор, тому управління в грі здійснювалося шляхом набору тексту. Команди для зміни курсу або стрільби вводилися в градусах: 0 означав праву сторону, 90 - верх, 270 - низ. Кнопки також могли використовуватися зі «стрілками». До терміналів були підключені монохромні дисплеї з дозволом 512x512 пікселів; для відображення графіки можна було завантажувати спеціальний набір символів.

Першим 3D шутером можна вважати Maze War - відеогра 1973 року, спочатку створена для Imlac PDS-1 Стівом Коллі в Дослідницькому центрі Еймса NASA. Поряд з Empire і Spasim стала однією з попередниць сучасних шутерів від першої особи, а також першою грою з режимом deathmatch.

У Maze War гравці переміщаються по лабіринту; доступна можливість переміщення вперед, назад, повертатися направо і наліво (кожен раз на 90 °), а також заглядати в дверні отвори. У грі використовується проста тайловая графіка - таким чином, гравець переміщається по невидимим квадратах. Інші учасники гри представлені на екрані у вигляді очних яблук. При появі суперника на екрані, гравець може стріляти в нього. За кожне вбивство нараховуються очки, а за кожну смерть - знімаються. У деяких версіях Maze War (наприклад, портована версія для X11) були впроваджені чит-коди, які дозволяли бачити місце розташування інших гравців. Також в деяких версіях в лабіринті могла іноді з'явитися качка.

Ігрові движки

Ігри ускладнювалися це призвело до того, що стали з'являтися ігрові движки - структури полегшують розробку ігор. Інструментарій, званий ігровими движками, створений для спрощення і прискорення розробки ігор, щоб не писати все «з нуля». В даному контексті буде фігурувати кілька понять движків, але найчастіше це ігрові та графічні. Важливо розуміти різницю між

графічним движком, ігровим движком і допоміжної бібліотекою ігрового движка. Ігровий движок - це той модуль гри, який включає в себе ігрову логіку. Наприклад, гра Pac-Man, крім всього іншого, містить код, який отрісовиває частково заповнений жовтий коло - головного героя (відноситься до графічному движку); і код, який збільшує бали, коли гравець з'їдає мисливця-примари, жовті точки, бонуси та інше (відноситься до ігрового движку). 1979 рік - «ZIL» Перший в світі ігровий движок розроблений компанією Infocom.

1987 рік - Freescape Перший 3-D движок розроблений компанією Incetive Software.

Unreal Engine

Багато движків створювалися для однієї єдиної гри, що робило неможливим використовувати цей движок в подальшому. Такий стан було збитковим, однак в 1998 році компанія Epic Games випускає движок став основою для сотень ігор, і при всій простоті був неймовірно потужною середовищем розробки. Написаний на мові C ++, движок дозволяє створювати ігри для більшості операційних систем і платформ: Microsoft Windows, Linux, Mac OS і Mac OS X; консолей Xbox, Xbox 360, PlayStation 2, PlayStation 3, PSP, PS Vita, Wii, Dreamcast, GameCube і інших, а також на різних портативних пристроях, наприклад, пристроях Apple (iPad, iPhone), керованих системою iOS і інших. (Вперше робота з iOS була представлена в 2009 році, в 2010 році продемонстровано роботу движка на пристрої з системою webOS).

Для спрощення портування движок використовує модульну систему залежних компонентів. Підтримує різні системи рендеринга: Direct3D, OpenGL, Pixomatic, відтворення звуку: EAX, OpenAL, DirectSound3D, засоби голосового відтворення тексту, розпізнавання мови, модулі для роботи з мережею та підтримуваних пристроїв введення.

Для гри по мережі підтримуються технології Windows Live, Xbox Live, GameSpy і інші, включаючи до 64 гравців (клієнтів) одночасно. Таким чином, движок адаптували і для застосування в іграх жанру MMORPG (один із прикладів: Lineage II).

Всі елементи ігрового движка представлені у вигляді об'єктів, що мають набір характеристик, і класу, який визначає доступні характеристики. У свою чергу, будь-який клас є «дочірнім» класом object. Серед основних класів і об'єктів можна виділити наступні:

Актор (actor) - батьківський клас, що містить всі об'єкти, які мають відношення до ігрового процесу і мають просторові координати.

Пішак (pawn) - фізична модель гравця або об'єкта, керованого штучним інтелектом. Назва походить від англ. pawn - той, ким маніпулюють (або пішак, тому такий об'єкт без будь-якої моделі виглядає як пішак). Метод управління описаний спеціальним об'єктом, такий об'єкт називається контролером. Контролер штучного інтелекту описує лише загальна поведінка пішаки під час ігрового процесу, а такі параметри як «здоров'я» (кількість пошкоджень, після яких пішак перестає функціонувати) або, наприклад, відстань, на якому пішак звертає увагу на звуки. задаються для кожного об'єкта окремо.

Світ, рівень (world, game level) - об'єкт, що характеризує загальні властивості «простору», наприклад, силу тяжіння і туман, в якому розташовуються всі актори. Також може містити в собі параметри ігрового процесу, як, наприклад, ігровий режим, для якого призначений рівень.

Для роботи з простими і, як правило, нерухомими елементами ігрового простору (наприклад, стіни) використовується двійкове розбиття простору - все простір ділиться на «заповнене» і «пусте». У «порожній» частини простору розташовуються всі об'єкти, а також тільки в ній може знаходитися «точка спостереження» при відображенні сцени. Можливість повного або часткового приміщення об'єктів в «заповнену» частину простору не виключається, однак може призвести до неправильної обробці таких об'єктів (наприклад, розрахунок фізичного взаємодії) або неправильної відтворення в разі приміщення туди «точки спостереження» (наприклад, ефект «залу дзеркал») . Все пішаки, які потрапляють в «заповнену» частина простору, відразу «гинуть».

Зонування. В камеру не потрапляє один портал (пунктирна лінія) червоної зони, тому об'єкти в ній не обробляються зовсім.

Поверхня (surface) є основним елементом двійкового дерева простору. Ці елементи створюються на межі перетину між «заповненої» і «порожній» частинами простору. Група елементів двійкового дерева простору називається нодом (node, рус. Вузол). Цей термін, як правило, вживається в контексті node count - кількість нодов на екрані або в ігровому просторі взагалі. Кількість нодов, одночасно видимих на екрані впливає на продуктивність при промальовуванні сцени. Якщо якийсь нод не потрапляє на екран або перекривається цілком іншими нодамі, він не обраховується - це служить для підвищення продуктивності, особливо в закритих просторах. Розбиття всього простору на групи нодов називається зонуванням.

Для цього іноді використовуються портали - невидимі поверхні, які служать для того щоб вручну розділити великий нод на два менших. Крім порталів використовуються антіпортали.

Опис «заповнених» і «порожніх» частин простору виконується за допомогою набору замкнутих тривимірних об'єктів, складених з не перетинаються поверхонь - Брашей (brush, рус. Кисть). Цей принцип побудови простору називається конструктивної суцільний геометрією. Геометрія може бути «адитивної» (весь простір спочатку «пусте») і «вичітательной» (спочатку заповнений матерією простір).

Браши діляться на три типи:

Суцільні (solid) - повноцінно беруть участь в довічним розбитті простору.

Адитивні (additive) - «заповнюють» двоичное простір.

Вичітательние (substractive) - «вирізають» обсяги в просторі.

Полу-суцільні (semi-solid) - не впливають безпосередньо на двійкове дерево простору, однак впливають на її фізичну модель. Можуть тільки «заповнювати» простір. Служать для створення «невидимих» перешкод, а також зниження числа полігонів і нодов.

Порожні (non-solid) - тільки створюють поверхні, не впливають на двійкове дерево простору. Використовуються переважно для створення обсягів (volume) - частина простору, яка має властивості, відмінними від властивостей

ігрового світу. Обсяги мають пріоритет, властивості обсягу з великим пріоритетом застосовуються до знаходяться в ньому акторам. Ігровий світ завжди має мінімальний пріоритет. За допомогою обсягів можна змінити гравітацію, в'язкість, туман тощо. Обсяги, починаючи з версії движка Unreal Engine 2, використовуються для створення води (але не водної поверхні).

Індустрія комп'ютерних ігор

Індустрія комп'ютерних ігор зародилася в середині 1970-х років як рух ентузіастів і за кілька десятиліть виросла з невеликого ринку в мейнстрім з річним прибутком в 9,5 мільярда доларів в США в 2007 році і 11,7 мільярда в 2008 році (за щорічними звітами ESA). На ринку працюють як великі гравці, так і і невеликі фірми і стартапи, а також незалежні розробники і спільноти.

Сучасні персональні комп'ютери дали безліч нововведень ігрової індустрії. До числа найбільш значущих відносять звукові і графічні карти, CD- і DVD-приводи, Unix і центральні процесори.

Звукові карти спочатку були розроблені для інтегрування якісного цифрового звуку в комп'ютерні ігри, і тільки потім звукове обладнання було вдосконалено під потреби меломанів.

Графічні карти, які на зорі комп'ютерної ери еволюціонували в напрямку збільшення кількості підтримуваних кольорів, пізніше стали розвиватися для апаратної підтримки графічних інтерфейсів користувача (англ. GUI) та ігор. Для GUI потрібно збільшення дозволу екрану, а для ігор - прискорення тривимірної графіки.

Спочатку CD і DVD були розроблені як недорогий і досить надійний спосіб зберігання і розповсюдження будь-яких даних. Згодом, коли ці технології стали застосовуватися в комп'ютерних іграх, почалося їх розвиток в бік збільшення швидкості читання даних.

Сучасні ігри - одні з найбільш вимогливих додатків на ПК. Багато потужні комп'ютери купуються геймерами, які потрібні для запуску новітніх ігор, в яких використовуються самі передові технології. Таким чином, ігрова індустрія тісно пов'язана з індустрією виробництва центральних процесорів і інші

компонентів ПК, так як гри часто вимагають більш високих апаратних потужностей, ніж бізнес-додатки.

Таким чином ігрова індустрія - одна з найперспективніших галузей інформаційних технологій. Багато людей подумали, а чому б і їм не зайнятися створенням комп'ютерних ігор?

Інді проекти

Інді-гра-«незалежна комп'ютерна гра» - комп'ютерна гра, створена окремим розробником або невеликим колективом без фінансової підтримки видавця комп'ютерних ігор. Поширення здійснюється за допомогою каналів цифрової дистрибуції. Масштаб явищ, пов'язаних з інді-іграми, відчутно зростає з другої половини 2000-х років, в основному через розвитку нових способів онлайн-дистрибуції і засобів розробки.

Загальноприйнятого визначення поняття «інді-гра» не існує. Але, найчастіше, інді-ігри мають деякі схожі особливості. Інді-ігри створюються окремими розробниками, невеликими колективами або маленькими незалежними компаніями. Також інді-ігри зазвичай не такі масштабні, як масові ігри з повним фінансуванням. Розробники інді-ігор, як правило, не мають фінансової підтримки від видавця (так як вони вважають за краще найменш ризикові гри з високим бюджетом), і зазвичай мають невелику бюджетом, або не володіють нею взагалі. З огляду на свою незалежність інді-розробники не мають операційних обмежень з боку видавців або творчих обмежень і не потребують схвалення видавця, що є обов'язковим для розробників масових ігор. Як наслідок, рішення геймдизайнера також не обмежуються бюджетом проекту. Більш того, чим менше колектив, тим яскравіше виражається індивідуальність конкретного розробника.

Створення ігор без програмування

У 2011 році США визнала гри видом мистецтва. Ігри несуть в собі якусь думку і ідею, яку автор намагається до нас донести; у гри є сюжет оповідає нам про долю головного героя, який показує нам його формування, становлення, який знайомить нас зі світом героя. Все це притаманне іншим видам

загальноприйнятого мистецтва: книгам, фільмам, музиці, картинам, скульптурі. За фактом гри дозволяю відчути, буквально обмацати і відчути кожну деталь і ланка в світі головного персонажа, як це зроблено в книгах, і при цьому насолоджуватися великими панорамами подій, бачити у всій красі світ, як це було реалізовано в кіно. Але на відміну від цих видів мистецтва в гру додається такий елемент, як гемплей. Ви вже не просто дивіться на головного героя з великого екрану, або читаєте про те, що він робив зі сторінок книги, а самі стаєте цим героєм, ви вершите ті подвиги, що призначалися персонажу, ви вирішуєте, що робити, ви тепер герой!

У зв'язку з цим створенням комп'ютерних ігор зайнялися люди мистецтва, які мало розуміли в програмуванні, так необхідному для геймдевелопінга.

Для таких людей були створені до остраху прості і в той же час потужні середовища розробок.Однією з таких є Unity.

Unity - це інструмент для розробки двох-і тривимірних додатків та ігор, що працює під операційними системами Windows і OS X. Створені за допомогою Unity програми працюють під операційними системами Windows, OS X, Windows Phone, Android, Apple iOS, Linux, а також на ігрових приставках Wii, PlayStation 3, PlayStation 4, Xbox 360, Xbox One. Є можливість створювати додатки для запуску в браузерах за допомогою спеціального модуля Unity (Unity Web Player), а також за допомогою реалізації технології WebGL. Раніше була експериментальна підтримка реалізації проектів в рамках модуля Adobe Flash Player, але пізніше команда розробників Unity прийняла складне рішення щодо відмови від цього.

Додатки, створені за допомогою Unity, підтримують DirectX і OpenGL. Активно движок використовується як великими розробниками (Blizzard, EA, QuartSoft, Ubisoft), так і девелоперами Indie-ігор (наприклад, ремейк Мор. Утопія (Pathologic), Kerbal Space Program, Slender: The Eight Pages, Slender: The Arrival, Surgeon Simulator 2013 і т. п.) в силу наявності безкоштовної версії, зручного інтерфейсу і простоти роботи з движком.

Редактор Unity має простий Drag & Drop інтерфейс, який легко налаштовувати, що складається з різних вікон, завдяки чому можна проводити налагодження гри прямо в редакторі. Движок підтримує три сценарних мови: C #, JavaScript (модифікація), Boo (діалект Python). Редактор підтримує DirectX 11 і HDR. Розрахунки фізики виробляє фізичний движок PhysX від NVIDIA.

Проект в Unity ділиться на сцени (рівні) - окремі файли, що містять свої ігрові світи зі своїм набором об'єктів, сценаріїв, і налаштувань. Сцени можуть містити в собі як, власне, об'єкти (моделі), так і порожні ігрові об'єкти - об'єкти, які не мають моделі («пустушки»). Об'єкти, в свою чергу містять набори компонентів, з якими і взаємодіють скрипти. Також у об'єктів є назва (в Unity допускається наявність двох і більше об'єктів з однаковим назвами), може бути тег (мітка) і шар, на якому він повинен відображатися. Так, у будь-якого об'єкта на сцені обов'язково присутній компонент Transform - він зберігає в собі координати місця розташування, повороту, і розмірів об'єкта по всіх трьох осях. У об'єктів з видимої геометрією також за замовчуванням присутній компонент Mesh Renderer, що робить модель об'єкта видимою.

До об'єктів можна застосовувати колізії (в Unity т. Н. Колайдери - collider). Існує кілька типів коллайдеров:

Character controller - вид фізичної моделі, створений спеціально під використання його для ігрових персонажів;

Box collider (фізична модель утворює куб, в який потрапляє вся модель об'єкта);

Sphere collider (фізична модель утворює сферу, в яку потрапляє вся модель об'єкта);

Capsule collider (фізична модель утворює капсулу, в яку потрапляє модель об'єкта. На відміну від попереднього типу розміри можна змінювати і по одній, і по трьох осях відразу);

Mesh collider (фізична модель повністю повторює реальну геометрію об'єкта);

Wheel collider (фізична модель колеса);

Terrain collider - тип фізичної моделі, створений спеціально для використання на об'єкті типу Terrain - земля, що генерується редактором Unity з можливостями скульптінга і фарбування місцевості.

Також Unity підтримує фізику твердих тіл і тканини, а також фізику типу Ragdoll (тряпічная лялька). У редакторі є система успадкування об'єктів; дочірні об'єкти будуть повторювати всі зміни позиції, повороту і масштабу батьківського об'єкта. Скрипти в редакторі прикріплюються до об'єктів у вигляді окремих компонентів.

При імпорті текстури в Unity можна згенерувати alpha-канал, mip-рівні, normal-map, light-map, карту відображень, проте безпосередньо на модель текстуру прикріпити не можна - буде створено матеріал, з яким буде призначений шейдер, і потім матеріал прикріпиться до моделі. Редактор Unity підтримує написання і редагування шейдеров. Редактор Unity має компонент для створення анімації, але також анімацію можна створити попередньо в 3D-редакторі та імпортувати разом з моделлю, а потім розбити на файли.

Крім порожнього ігрового об'єкта і моделей, на сцену можна додавати ще такі об'єкти типу GameObject:

Система частинок;

Камера;

GUI текст;

GUI текстура;

3D текст;

Точкове світло;

Спрямований світло;

Освітлення території;

Джерело світла, що імітує сонце;

Стандартні примітиви;

дерева;

Terrain (земля).

Unity 3D підтримує систему Level Of Detail (скор. LOD), суть якої полягає в тому, що на далекій відстані від гравця високодеталізовані моделі замінюються на менш деталізовані, і навпаки, а також систему Occlusion culling, суть якої в тому, що у об'єктів, НЕ попадающіх в поле зору камери не візуалізується геометрія і колізія, що знижує навантаження на центральний процесор і дозволяє оптимізувати проект. При компіляції проекту створюється виконуваний (.exe) файл гри (для Windows), а в окремій папці - дані гри (включаючи всі ігрові рівні і спільні бібліотеки).

Движок підтримує безліч популярних форматів, таких як:

.3ds, .max, .obj, .fbx, .dae, .ma, .mb для тривимірних моделей;

.mp3, .wmv, .ogg для звукових файлів;

.bmp, .gif, .png, .tga, .psd, .tif, .dds для зображень;

.mov, .ovg для відеофайлів.

Моделі, звуки, текстури, матеріали, скрипти можна запаковувати в формат.unityassets і передавати іншим розробникам, або викладати у вільний доступ. Цей же формат використовується у внутрішньому магазині Unity Asset Store, в якому розробники можуть безкоштовно і за гроші викладати в загальний доступ різні елементи, потрібні при створенні ігор. Щоб використовувати Unity Asset Store, необхідно мати акаунт розробника Unity. Unity має всі потрібні компоненти для створення мультиплеєра. Також можна використовувати відповідний користувачеві спосіб контролю версій. Наприклад, Tortoise SVN або Source Gear.

Проте Unity вимагає не слабку підготовку і починаючому буде дуже важко створити гру в поодинці на такому движку. Тому були створені двигуни не потребують програмування зовсім, де все тримається на виборі властивостей об'єктах, як наприклад Construct 2

Але він дуже і дуже обмежений, тому що не можна ставити інші властивості, крім передбачених.

Проте наступний движок дає повноту можливості розробки 2-D ігор без обмежень на поставлені властивості, при збереженні відносної простоти користування - Game Maker

Maker: Studio - один з найвідоміших конструкторів ігор. Написаний на Delphi. Доступний для ОС Windows, 7-я версія програми також існувала в версії для Mac. Провідний розробник - Марк Овермарс.

Система розрахована в основному на створення двомірних (2D) ігор будь-яких жанрів. Також підійде для створення різних презентацій і т. П. Починаючи з 6-ї версії з'явилася обмежена можливість працювати з 3D.

Може бути рекомендований для вивчення програмування. Будучи професором Утрехтського університету Марк Овермарс почав розробляти Game Maker як навчальний посібник для своїх студентів.

Game Maker поширюється на умовах Shareware, безкоштовна версія обмежена в функціональності, а при запуску відкомпільованих в ній ігр показується логотип програми.

Однак Game Maker Studio після реєстрації та оновлення до Standart версії (Безкоштовно) майже не має обмежень і не показує логотип студії при запуску гри.

Створення гри в Game Maker не вимагає попереднього знайомства з яким небудь з мов програмування.

Інтерфейс Game maker об'єднує в собі редактори спрайтів, об'єктів, кімнат, скриптів, а також тайм-лайнів (послідовностей дій з прив'язкою за часом) і шляхів (маршрутів) руху.

Гра в Game maker будується як набір ігрових об'єктів. За їх зовнішній вигляд відповідають спрайт, а поведінка задається шляхом опису реакцій на події. Для цього можна використовувати графічне представлення програм (близьке до блок-схемами) у вигляді послідовності іконок-дій. Програмування за допомогою дій відбувається в режимі drag-n-drop. Наприклад, для того щоб почати умовний оператор, потрібно перетягнути на панель дій восьмикутник з іконкою, що позначає тип перевірки, а потім, можливо, ввести будь-які

значення в форму, що з'явилася. Для більш просунутих користувачів є скриптова мова GML схожий на JavaScript, є можливість створення власних бібліотек дій, використовуючи Library Maker.

Поняття об'єкта в GameMaker в основному відповідає поняттю класу в об'єктно-орієнтованому програмуванні, об'єкти можуть успадковувати один від одного. Примірники об'єктів можуть бути розміщені в ігровому просторі за допомогою редактора кімнат, або ж створені динамічно. Якщо в поточній кімнаті існує тільки один екземпляр об'єкта, до нього можна звертатися, використовуючи родове ім'я об'єкта, класу, якщо ж таких екземплярів кілька, для звернення до конкретного екземпляру ми повинні знати його числовий ідентифікатор, використовуючи його в якості посилання на об'єкт.

Мова GML включає в себе засоби завантаження і використання зовнішніх динамічних бібліотек, що дозволяє розширювати Game Maker процедурами і функціями, написаними на інших мовах. Зовнішні DLL, разом з gml обв'язкою можуть бути зібрані в пакет розширення GameMaker.

У даного движка є цілий ряд переваг і недоліків:

+ Кроссплатформенность;

+ Гнучка цінова категорія, базова версія Game Maker: Studio абсолютно безкоштовна;

+ Власний спрощений мова програмування Game Maker Language (GML);

+ Інтеграція з Steam;

+ Підтримка безлічі інтернет-майданчиків «з коробки» (Developer Services Portal);

- погано оптимізований для великих ігор;

- незважаючи на можливість роботи з 3D, в Game Maker вона вкрай незручна;

- сам Game Maker Language (GML) має ряд помітних недоліків, що, тим не менш, не завадить початківцям розробникам.

1.2 Що таке DirectX?

Уявіть собі ситуацію: ви розробляєте гру і як розробник (не кажучи вже про видавця) бажаєте, щоб гра могла працювати на якомога більшій кількості комп'ютерів. Але для цього вам необхідно врахувати всі мислимі і немислимі відео-, звукові і мережеві карти, керма, миші, клавіатури і ще цілу купу різного устаткування. Набагато легше створити певний стандарт, завдяки якому написання коду, наприклад, для пристроїв введення, можна зробити один раз, працювати він буде завжди і скрізь. До цього і прагне корпорація Microsoft вже багато років, і з кожною новою версією DirectX їй це вдається все краще і краще. DirectX - це мультимедійна бібліотека, що дозволяє безпосередньо працювати з апаратним забезпеченням комп'ютера в обхід традиційних засобів платформи Win32. Тісна взаємодія DirectX і обладнання з драйверами, написаними виробником даного обладнання, дають можливіість повністю відволіктися від апаратної частини і зосередити свою увагу на створенні правильного коду. Вся DirectX ділиться на компоненти, що відповідають за ту чи іншу частину роботи бібліотеки:

- DirectX Graphics;
- DirectInput;
- DirectMusic;
- DirectSound;
- DirectPlay;
- DirectShow;
- DirectSetup.

На рис. 1 представлена загальна схема взаємодії DirectX 9 з апаратним забезпеченням комп'ютера.

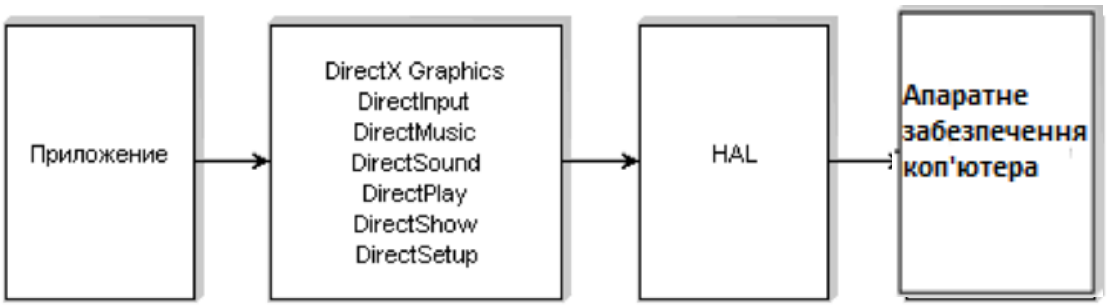

Рис. 1

Direct3D

Direct3D - це найпотужніший і самий головний компонент DirectX 9, відповідальний за отрисовку тривимірної графіки, накладення на об'єкт текстур, за вершинні і піксельні шейдери і багато іншого. За допомогою Direct3D існує можливість вбудовувати тривимірну графіку в додатках для платформи Windows. Маючи в своєму розпорядженні великий набір різних функцій, структур, макросів, прапорів, розробник практично відсунутий від апаратної частини комп'ютера і зосереджений на програмному коді. Сам же програмний код, а точніше безліч функцій тривимірної графіки, дуже легко вписуються і взаємодіють з додатком. Найголовнішою складовою Direct3D є процес рендеринга (rendering). Рендерінг- це візуалізація тривимірного об'єкту на екрані монітора. Тут важливо розуміти, що будь-який об'єкт зберігається в відео- або системної пам'яті комп'ютера, як розібраний на частини конструктор, де кожна з частин відповідає за ту чи іншу складову об'єкта. наприклад, робот з конструктора спочатку складається з голови, ніг, рук, туловища, точно так само і об'єкт розбитий на частини, що складаються з текстур, світла і матеріалу, полігонів і багато чого іншого. Щоб побачити цілого робота в конструкторі, його потрібно зібрати з частин, щоб побачити об'єкт на екрані монітора, він повинен пройти через процес рендеринга, який зробить збірку об'єкта. Звичайно, насправді я тут трохи спрощую загальну концепцію, також під об'єктом мається на увазі не тільки якась окремо взята тривимірна модель, але і весь навколишній її світ. В Direct3D це зветься сценою.

Весь процес рендеринга складається з декількох механізмів - тесселяції, трансформації, освітлення і растеризації. За допомогою цих механізмів або, як кажуть, конвеєра рендеринга, досягається візуалізація всієї сцени на екрані.

Тесселяція (tessellation) - це механізм розбиття поверхні на полігони. Полігон - це певна площа в тривимірному просторі, обмежена кутами. В Direct3D полігоном зазвичай є трикутник.

Трансформація (transformation) - це перетворення координат об'єкту з помошью матриць, які виробляють обертання, перенос, масштабування і інші різні матричні перетворення об'єкта.

Освітлення (lighting) - пряма аналогія нашого світу, де всі предмети мають колір, який в свою чергу залежить від джерел освітлення.

Растеризация (rasterizer) - це найскладніший механізм, який, по суті, і малює всю сцену на екрані монітора, представляючи її на піксельному рівні, здійснюючи текстурирование, 2-6уферізацію, затінення, альфа-змішування, приховування невидимих поверхонь і багато іншого.

Побудова сцени в Direct3D

Принципу побудови сцени, який умовно можна розбити на кілька частин.

1. Створення віконного програми. Створивши вікно, в будь якому вибраному вами режимі і вигляді, ви створюєте собі робочу поверхню, куда будете послідовно інтегрувати свій код і функції DirectX.

2. Ініціалізація Direct3D. Тут ви створюєте ряд покажчиків на необхідні вам інтерфейси, инициализируется Direct3D 9, задаються параметри уявлення і створюється пристрій Direct3D. Пристрій Direct3D -це, якщо можна так сказати, апаратна і програмна частина вашого комп'ютера. Тобто ви задаєте розширення екрана, частоту зміни кадрів, дізнаєтеся інформацію про відеокарту, встановлюєте, як відеоадаптер буде обробляти запропоновану йому інформацію, задаєте формат заднього буфера, встановлюєте параметри буфера глибини, взагалі все те, що стосується налаштування апаратної та програмної частин вашого компьютера.

3. Створення об'єкта. Кожен об'єкт, яким би він великим не був, складається з полігонів. Зазвичай це трикутники. Будь-трикутник складається з трьох кутів або точок. У Direct3D оперують поняттям вершини. Вершина задається координатами в просторі по трьох осях X, У і Z. Знаючи все

це, а також маючи уявлення про тригонометрії, можна побудувати практично будь-який об'єкт. Ставлячи формат вершин в залежності від того, чи використовуєте ви матричні перетворення для рендеринга об'єкта або користуєтеся перетвореним форматом вершин, для подання 2D-об'єкта на екрані, ви тим самим створюєте об'єкт.

4. Освітлення, матеріал і текстура. Реально сцену можна уявити на екрані, якщо використовувати освітлення і матеріал. Створивши різні джерела світла, наприклад, лампочку, сонце, вогонь, і задавши матеріал для об'єкта, який визначає відображення джерела світла, ви додасте реалізму мальованої сцені. Ну, і наклавши текстуру, наприклад, у вигляді паркету на пол, ви досягнете абсолютної ідентичності нашого світу.

5. Візуалізація об'єкта. Можна промальовувати сцену на екрані, задаючи різні значення для всіх компонентів конвеєра рендеринга. Звичайно, все це так само для наочності розділені на умовні складові, але загальний принцип, я думаю тепер зрозумілий. Розглянемо, з яких інтерфейсів складається Direct3D 9.

Інтерфейси Direct3D 9

Direct3D 9 містить найбільшу кількість різних інтерфейсів отвечаюшіх за текстури, вершинні і піксельні шейдери, буфер вершин, індексний буфер, інтерфейс пристрою Direct3D 9 і т. д. Все інтерфейси ми розглянути не в силах, тому розберемо тільки безпосередньо зустрічаються в книзі:

- IDirect3D9;
- IDirect3DDevice9;
- IDirect3DVertexBuffer9;
- IDirect3DIndexBuffer9;
- IDirect3DTexture9;
- IDirect3DVertexShader9.

IDirect3D9 - найголовніший інтерфейс, з нього успадковуються всі інші інтерфейси. Це перший об'єкт, який ви повинні створити, і тільки потім можна отримати доступ до всіх інших інтерфейсів і функцій. Об'єкт інтерфейсу створюється за допомогою функції Direct3DCreate9().

IDirect3DDevice9 - після того як створений об'єкт основного інтерфейсу Direct3D9, створюється інтерфейс пристрою Direct3D. Як я вже казав, це, по суті, певний набір необхідних налаштувань апаратної і програмної складових комп'ютера, що створюють пристрій Direct3D, на основі якого малюється вся сцена. Якщо ми говоримо про графік, то очевидно, що пристрій Direct3D представлятиме відеоадаптер. цей інтерфейс завжди створюється другим, за допомогою функції IDirect3D9::CreateDevice. І, звичайно ж, даний інтерфейс містить ще ряд функцій, які здійснюють операції з пристроєм Direct3D, також це повною мірою стосується і інших інтерфейсів.

IDirect3DVertexBuffer9 - інтерфейс, за допомогою якою створюється буфер вершин. Кожен об'єкт складається з трикутників, заданих точками в просторі (вершин). Зрозуміло, що весь об'єкт складається з певного набору вершин, і для того, щоб всі ці вершини було зручно зберігати, створюється буфер вершин, що є в загальному вигляді масивом даних. Буфер вершин створюється при ДОПОМОГИ функції IDirect3DVertexBuffer9::CreateVertexBuffer.

Direct3DIndexBuffer9 - використовуючи цей інтерфейс, ви можете створити індексний буфер, що дозволяє проіндексувати велику кількість вершин об'єкта. Наприклад, ви маєте прямокутник, що складається з двох трикутників, де принаймні чотири вершини (по дві на кожен з двох кутів)мають

однакові координати, тобто повторюються. За допомогою індексного буфера можна проіндексувати вершини і уникнути повторення. Може бути, звичайно, для трикутника це і не актуально, але для куба індексація вершин доведеться до речі. При необхідності індексний буфер можна створити ЗА ДОПОМОГОЮ функції IDirect3DIndexBuffer9:: CreateindexBuffer.

IDerect3DTexture9 - як видно з назви, цей інтерфейс відповідає затекстури. У його склад входить багато функцій, за допомогою яких ви можете накладати текстури на об'єкт, одну на іншу і т. д.

IDerect3DVertexShader9 - цей інтерфейс, як мені здається, відповідає за майбутнє ігрової індустрії, по крайней мере, на кілька найближчих років точно. Справа в тому, що механізм трансформації і освітлення в більш серйозних додатках потихеньку здає свої позиції, оскільки має свої обмеження у використанні, і йому на зміну прийшли вершинні шейдери, що поліпшують якість графіки. Верховий шейдер - це програма, написана на мові, подібному асемблеру, і інтегрована в створюване додаток, що дозволяє в реальному часі досягти більш якісного рівня графіки.

1.3 Python як мова програмування

Мова Python неймовірно ефективний: ваші програми роблять більше, ніж багато інших мов, в меншому обсязі коду. Синтаксис Python також дозволяє писати «чистий» код. Ваш

код буде легко читатися, у вас буде менше проблем з налагодженням і розширенням програм в порівнянні з іншими мовами.

Чи варто вчити Python?

Думаю, ні для кого не секрет, що буквально ще 5-6 років тому, першою мовою для вивчення програмування в будь-якій школі був Pascal. Pascal чудовий мову своєї епохи, але, на жаль чи на щастя, він своє віджив і тепер Python як перша мова програмування це розумний вибір для кожного початківця програмістаю.

Переваги мови програмування Python

По-перше, важливо знати, що дана мова програмування зараз затребуваний у багатьох підприємствах, особливо це виражено в Москві і Санкт-Петербурзі. Так що якщо вам пощастило жити в цих містах, то з цією мовою можна пошукати дуже цікаву і високооплачувану роботу.

По-друге, це розвивається мова програмування, різні зміни в ньому відбуваються раз в два-три роки, а це дуже хороший показник для мови програмування.

По-третє, це відносно проста мова. Як таке може бути? А от не повірите, буває! За фактом, для оволодіння цією мовою достатньо вміти розуміти текст англійською мовою. Якщо ви це вмієте, то більшість функцій мови будуть вам зрозумілі. До того ж, якщо почитати відгуки програмістів про мову Python, то можна помітити, що ця мова - улюбленець майже кожного!

Програмісти просто обожнюють цю мову за стислість і простоту коду. Там де в мові JavaScript або C ++ вам буде потрібно написати дві-три сторінки, в Pythone ви укладетеся всього в одну!

А ще дуже важливою особливістю мови Python є те, що він застосовується для Web-розробок. Причому він використовується не як звичайну мову, а виконує одну з цікавих функцій. Python для web-розробок застосовується в тих випадках, коли інші мови не справляються! Адже це дуже цікавий і цікавий факт. У даній сфері у мови своя власна, особиста ніша!

Python використовується для різних цілей: для створення ігор, побудови вебпріложеній, рішень бізнес-завдань і розробки внутрішніх інструментів для всіляких цікавих проектів. Python також широко застосовується в науковій області для теоретичних досліджень і вирішення прикладних завдань. Втім, однією з найважливіших причин для використання Python для мене залишається співтовариство Python, що складається з неймовірно різних і доброзичливих людей. Спільнота грає виключно важливу роль в програмуванні, тому що програмування не є суто індивідуальною справою. Багатьом з нас, навіть найдосвідченішим програмістам, доводиться звертатися

за порадою до колег, які вже вирішували схожі завдання. Існування дружного, доброзичливого спільноти допомагає вирішувати завдання.

Python в різних операційних системах

Python є крос-платформних мовою програмування; це означає, що він працює у всіх основних операційних системах. Будь-яка програма на мові Python, написана вами, повинна виконуватися на будь-якому сучасному комп'ютері зі встановленою підтримкою Python. Втім, способи настройки Python для різних операційних систем злегка відрізняються.

Python в системі Linux

Системи сімейства Linux орієнтовані на програмістів, тому підтримка Python вже встановлена на більшості комп'ютерів Linux. Люди які займаються розробкою і супроводом Linux, очікують, що в якийсь момент ви займетеся програмуванням, і всіляко сприяють цьому. з цієї причини для переходу до програмування вам майже нічого не доведеться встановлювати, а кількість необхідних налаштувань буде мінімальним.

Перевірка версії Python

Відкрийте термінальне вікно, запустивши додаток Terminal у вашій системі (В Ubuntu натисніть клавіші Ctrl + Alt + T). Щоб перевірити, чи встановлена підтримка Python у вашій системі, введіть команду python (з малої літери p). На екрані з'явиться інформація про те, яка версія Python у вас встановлена, і запрошення >>>, в якому можна вводити команди Python:

```
$ python
Python 2.7.6 (default, Mar 22 2014 року, 22:59:38)
[GCC 4.8.2] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Цей висновок повідомляє, що Python 2.7.6 в даний час є версією Python за замовчуванням, встановленої на даному комп'ютері. Натисніть Ctrl + D або введіть exit (), щоб вийти з запрошення Python і повернутися до запрошення терміналу. Щоб перевірити наявність Python 3, можливо, вам доведеться

вказати цю версію; отже, навіть при тому, що в якості версії за умовчанням в вихідних даних вказано Python 2.7, спробуйте ввести команду python3:

```
$ python3
Python 3.5.0 (default, Sep 17 2015 року, 13:05:18)
[GCC 4.8.4] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

З вихідних даних видно, що в системі також встановлена версія Python 3, так що ви зможете використовувати будь-яку з цих версій.

Філософія Python

Довгий час мова програмування Perl був наріжним каменем інтернетпрограммірованія. На перших порах функціонування багатьох інтерактивних сайтів було засновано на сценарії Perl. У той час співтовариство Perl керувалося девізом: «Це можна зробити декількома способами». Якийсь часрозробникам подобався такий підхід, тому що гнучкість, притаманна мови, дозволяла вирішувати багато завдань різними способами. Подібний підхід був допустимо при роботі над власними проектами, але з часом стало ясно, що надмірна гнучкість ускладнює довгострокове супровід великих проектів. Було дуже важко, виснажливо і довго розбиратися в коді і намагатися зрозуміти, що ж думав інший розробник при вирішенні складного завдання. Досвідчені програмісти Python рекомендують уникати зайвих складнощів і застосовувати прості рішення там, де це можливо. Філософія спільноти Python виражена в нарисі Тіма Пітерса «The Zen of Python». Щоб переглянути цей короткий набір принципів написання хорошого коду Python, досить ввести команду import this в інтерпретаторі. Я не стану відтворювати всі принципи, але наведу кілька рядків.

```
>>> import this
The Zen of Python, by Tim Peters
```

‰Красиве краще, ніж потворне.

Програмісти Python вважають, що код може бути красивим і елегантним. У програмуванні люди займаються вирішенням завдань. програмісти завжди цінували добре спроектовані, ефективні і навіть гарні рішення. Згодом ви більше дізнаєтесь про Python, почнете писати більше коду, і коли-небудь ваш колега подивиться на екран вашого комп'ютера і скаже: «Ого, який красивий код! »

Просте краще, ніж складне.

Якщо у вас є вибір між простим і складним рішенням і обидва працюють, використовуйте просте рішення. Ваш код буде простіше в супроводі, а у вас і інших розробників буде менше проблем з оновленням цього коду в майбутньому.

Складне краще, ніж заплутане.

Реальність створює свої складності; іноді просте рішення задачі неможливо. В такому випадку використовуйте найпростіше рішення, яке працює.

Удобочитаемость має значення.

Навіть якщо ваш код складний, він повинен нормально читатися. Працюючи над проектом,

вимагає написання складного коду, постарайтеся написати змістовні коментарі для цього коду.

Повинен існувати один - і бажано тільки один - очевидний спосіб зробити це.

Якщо запропонувати двом програмістам Python вирішити одну і ту ж задачу, вони повинні виробити схожі рішення. Це не означає, що в програмуванні немає місця для творчості. Навпаки! Але більша частина роботи програміста полягає в застосуванні невеликих, стандартних рішень для простих ситуацій в контексті великого, більш творчого проекту. Внутрішня організація ваших програм повинна виглядати логічно з точки зору інших програмістів Python.

Зараз краще, ніж ніколи.

Ви можете витратити весь залишок життя на вивчення всіх тонкощів Python і програмування в цілому, але тоді ви ніколи не закінчите жоден проект.

Не намагайтеся написати ідеальний код; напишіть код, який працює, а потім вирішите, чи варто доопрацювати його для поточного проекту або ж перейти на щось інше.

Стиль програмування

Не шкодуйте часу на те, щоб ваш код читався як можна простіше. зрозумілий код допомагає стежити за тим, що робить ваша програма, і спрощує вивчення вашого коду іншими розробниками. Програмісти Python виробили ряд угод по стилю, щоб весь код мав хоча б віддалено схожу структуру. Навчившись писати «чистий» код Python, ви зможете зрозуміти загальну структуру коду Python, написаного будь-яким іншим програмістом, хто виконує ті ж рекомендації.

Коли хто-небудь хоче внести зміни в мову Python, він пише документ PEP (Python Enhancement Proposal). Одним з найстаріших PEP є документ PEP 8 до рекомендацій по стильовому оформленню коду. PEP 8 має досить велику довжину, але більша частина документа присвячена більш складним програмним структурам, ніж ті, які зустрічалися вам до справжнього моменту.

Керівництво по стилю Python було написано з урахуванням того факту, що код читається частіше, ніж пишеться. Ви пишете свій код один раз, а потім починаєте читати його, коли переходите до налагодження. При розширенні функціональності програми ви знову витрачаєте час на читання свого коду. А коли вашим кодом починають користуватися інші програмісти, вони теж читають його.

Вибираючи між написанням коду, який простіше пишеться, і кодом, який простіше читається, програмісти Python майже завжди рекомендують другий варіант. Наступні поради допоможуть вам з самого початку писати чистий, зрозумілий код.

Відступи

PEP 8 рекомендує позначати рівень відступу чотирма пробілами. Використання чотирьох прогалин спрощує читання програми і при цьому залишає достатньо місця для декількох рівнів відступів в кожному рядку.

У програмах форматування тексту для створення відступів часто використовуються табуляції замість пробілів. Такий спосіб добре працює в текстових процесорах, але інтерпретатор Python приходить в замішання, коли табуляції змішуються з пробілами. У кожному текстовому редакторі є параметр конфігурації, який замінює натискання клавіші табуляції заданою кількістю прогалин. Звичайно, клавіша табуляції зручна, але ви повинні простежити за тим, щоб редактор вставляв в документ прогалини замість табуляцій.

Змішання табуляцій і прогалин у файлі може створити проблеми, сильно ускладнюють діагностику. Якщо ви думаєте, що в програмі табуляції змішалися з пробілами, пам'ятайте, що в більшості редакторів існує можливість перетворення всіх табуляцій в прогалини.

Довжина рядків

Багато програмістів Python рекомендують обмежувати довжину рядків 80 символами. Історично ця рекомендація з'явилася через те, що в більшості комп'ютерів в одному рядку терміналу містилося всього 79 символів. В даний час на екранах поміщаються куди довші рядки, але для застосування стандартної довжини рядка в 79 символів існують і інші причини. Професійні програмісти часто відкривають на одному екрані відразу кілька файлів; стандартна довжина рядка дозволяє бачити все рядки в двох або трьох файлах, відкритих на екрані одночасно. PEP 8 також рекомендує обмежувати коментарі 72 символами на рядок, тому що деякі службові програми, автоматично генеруючі документацію в великих проектах, додають символи форматування на початку кожного рядка коментаря.

Порожні рядки застосовуються для візуальної угруповання частин програми. Використовуйте порожні рядки для структурування файлів, але не зловживайте ними. Приклади, наведені в книзі, допоможуть вам виробити потрібний баланс. Наприклад, якщо в програмі п'ять рядків коду створюють

список, а потім наступні три рядки щось роблять з цим списком, два фрагмента доречно розділити порожній рядком. Проте між ними не варто вставляти три або чотири порожні рядки. Порожні рядки не впливають на роботу коду, але відображаються на його удобочитаемости. Інтерпретатор Python використовує горизонтальні відступи для інтерпретації сенсу коду, але ігнорує вертикальні інтервали.

Читання з файлу

Гігантські обсяги даних доступні в текстових файлах. У них можуть зберігатися погодні дані, соціально-економічна інформація, літературні твори та багато іншого. Читання з файлу особливо актуально для додатків, призначених для аналізу даних, але воно також може стати в нагоді в будь-який ситуації, що вимагає аналізу або зміни інформації, що зберігається у файлі. Наприклад, програма може читати вміст текстового файлу і переписувати його з форматуванням, розрахованим на відображення інформації в браузері.

Робота з інформацією в текстовому файлі починається з читання даних в пам'ять. Ви можете прочитати весь вміст файлу або ж читати дані по рядках.

Читання всього файлу

Для початку нам знадобиться файл з кількома рядками тексту. Нехай це буде файл з числом «пі» з точністю до 30 знаків, по 10 знаків на рядок:

pi_digits.txt

3.1415926535

8979323846

2643383279

Щоб випробувати ці приклади, введіть дані в редакторі і збережіть файл з ім'ям pi_digits .txt

Наступна програма відкриває цей файл, читає його і виводить вміст на екран:

file_reader.py

```
with open('pi_digits.txt') as file_object:
contents = file_object.read()
```

print(contents)

У першому рядку цієї програми багато заслуговує вашої уваги. Почнемо з функції open (). Щоб виконати будь-які операції з файлом - навіть просто вивести його вміст, - спочатку необхідно відкрити файл. Функція open () отримує один аргумент: ім'я файлу, що відкривається. Python шукає файл із зазначеним ім'ям в каталозі, в якому знаходиться файл поточної програми. В даному прикладі виконується програма file_reader .py, тому Python шукає файл pi_digits .txt в каталозі, в якому зберігається file_reader .py. Функція open () повертає об'єкт, представляє файл. В даному випадку open ('pi_digits.txt') повертає об'єкт, який представляє файл pi_digits .txt. Python зберігає цей об'єкт у змінній file_object, з якої ми будемо працювати пізніше в програмі.

Конструкція з ключовим словом with закриває файл після того, як потреба в ньому відпаде. Зверніть увагу: в цій програмі є виклик open (), але немає виклику close (). Файли можна відкривати і закривати явними викликами open () і close (); але якщо через помилку в програмі команда close () залишиться невиконаним, то файл не буде закритий. На перший погляд це не страшно, але некоректне закриття файлів може привести до втрати або псування даних. А якщо функція close () буде викликана занадто рано, програма спробує працювати з закритим (Тобто недоступним) файлом, що призведе до нових помилок. Не завжди можна заздалегідь визначити, коли потрібно закривати файл, але з наведеної конструкцією Python зробить це за вас. Вам залишається лише відкрити файл і працювати з ним так, як потрібно, сподіваючись на те, що Python закриє його автоматично в правильний момент. Після того як в програмі з'явиться об'єкт, який представляє файл pi_digits .txt, у другому рядку програми використовується метод read (), який читає весь вміст файлу і зберігає його вміст в одній довгій рядку в змінної contents. При виведенні значення contents на екрані з'являється весь вміст файлу:

3.1415926535
 8979323846
2643383279

Єдина відмінність між висновком і вихідним файлом - зайва порожній рядок в кінці виведення. Звідки вона взялася? Метод read () повертає її при читанні, якщо досягнуто кінець файлу. Якщо ви хочете видалити зайву порожній рядок, включите виклик rstrip () в команду print:

```
with open ('pi_digits.txt') as file_object:
contents = file_object.read ()
print (contents.rstrip ())
```

Нагадаємо, що метод rstrip ( ) видаляє всі пропуски в кінці рядка. Тепер висновок точно відповідає вмісту вихідного файлу:

```
3.1415926535
8979323846
2643383279
```

Шляхи до файлів

Якщо передати функції open () просте ім'я файлу, таке як pi_digits .txt, Python шукає файл в тому каталозі, в якому знаходиться файл, що виконується в даний момент (тобто файл програми .py).

У деяких випадках (в залежності від того, як організовані ваші робочі файли) відкривається файл може і не перебувати в одному каталозі з файлом програми. Наприклад, файл програми може знаходитися в каталозі python_work; в каталозі python_work створюється інший каталог з ім'ям text_files для текстових файлів, з якими працює програма. І хоча папка text_files знаходиться в python_work, проста передача open () імені файлу з text_files не підійде, тому що Python проведе пошук файлу в python_work і на цьому зупиниться; запитом не знайдено буде продовжений у вкладеному каталозі text_files. Щоб відкрити файли з каталогу, відмінного від того, в якому зберігається файл програми, необхідно вказати шлях – тобто наказати Python шукати файли в конкретному місці файлової системи.

Так як каталог text_files знаходиться в python_work, для відкриття файлу з text_files можна скористатися відносним шляхом. Відносний шлях наказує

Python шукати файли в каталозі, який задається щодо каталогу, в якому знаходиться поточний файл програми. В системі Linux і OS X це виглядає так:

with open ('text_files / імя_файла.txt') as file_object:

Цей рядок означає, що файл .txt слід шукати в каталозі text_files; вона передбачає, що каталог text_files знаходиться в python_work (так воно і є). У системахWindows в шляхах до файлів замість слеша (/) використовується зворотний слеш (\):

with open ('text_files \ імя_файла.txt') as file_object:

Також можна точно визначити місцезнаходження файлу у вашій системі незалежно від того, де зберігається виконувана програма. Такі шляхи називаються абсолютними і використовуються в тому випадку, якщо відносний шлях не працює. Наприклад, якщо каталог text_files знаходиться не в python_work, а в іншому місці (Скажімо, в каталозі з іменем other_files), то передати open () шлях 'text_files / filename.txt 'не вийде, тому що Python буде шукати вказаний каталог тільки всередині python_work. Щоб пояснити Python, де слід шукати файл, необхідно записати повний шлях.

Абсолютні шляхи зазвичай довше відносних, тому їх краще зберігати в змінних, які потім передаються open (). В Linux і OS X абсолютні шляхи виглядають так:

file_path = '/home/ehmatthes/other_files/text_files/імя_файла.txt'

with open (file_path) as file_object:

У Windows вони виглядають так:

file_path = 'C: \ Users \ ehmatthes \ other_files \ text_files \ імя_файла.txt'

with open (file_path) as file_object:

З абсолютними шляхами ви зможете читати файли з будь-якого каталогу вашої системи. Поки буде простіше зберігати файли в одному каталозі з файлами програм або в каталогах, вкладених в каталог з файлами програм (таких як text_files з розглянутого прикладу).

У процесі читання файлу часто буває потрібно обробити кожен рядок. Можливо, ви шукаєте якусь інформацію в файлі або збираєтеся якимось чином

змінити текст, наприклад при читанні файлу з метеорологічними даними ви обробляєте кожен рядок, у якій в описі погоди зустрічається слово «Сонячно». Або, припустимо, в новинах ви шукаєте кожен рядок з тегом заголовка і замінюєте її спеціальними елементами форматування.

Для послідовної обробки кожного рядка у файлі можна скористатися циклом for:

```
   file_reader.py
❶ filename = 'pi_digits.txt'

❷ with open(filename) as file_object:
❸     for line in file_object:
          print(line)
```

У точці 1 ім'я файлу, з якого читається інформація, зберігається в змінної filename. Це стандартний прийом при роботі з файлами: так як змінна filename не представляє конкретний файл (це всього лише рядок, яка повідомляє Python, де знайти файл), ви зможете легко замінити 'pi_digits.txt' ім'ям іншого файлу, з яким ви збираєтеся працювати. Після виклику open () об'єкт, який представляє файл і його вміст, зберігається в змінної file_object 2. Ми знову використовуємо синтаксис with, щоб доручити Python відкривати і закривати файл в потрібний момент. Для перегляду вмісту все рядки файлу перебираються в циклі for по об'єкту файлу 3.

На цей раз порожніх рядків виявляється ще більше:

3.1415926535

 8979323846

 2643383279

Порожні рядки з'являються через те, що кожен рядок в текстовому файлі завершується невидимим символом нового рядка. Команда print додає свій символ нового рядка при кожному виклику, тому в результаті кожен рядок завершується двома символами нового рядка: один прочитаний з файлу, а інший доданий командою print. Виклик rstrip () в команді print видаляє зайві порожні рядки:

filename = 'pi_digits.txt'

with open (filename) as file_object:

for line in file_object:

print (line.rstrip ())

Тепер висновок знову відповідає вмісту файлу:

3.1415926535

8979323846

2643383279

Робота з вмістом файла

Після того як файл буде прочитаний в пам'ять, ви зможете обробляти дані так, як вважаєте за потрібне. Для початку спробуємо побудувати один рядок з усіма цифрами з файлу без проміжних пропусків:

pi_string.py

filename = 'pi_digits.txt'

with open (filename) as file_object:

lines = file_object.readlines ()

```
❶ pi_string = ''
❷ for line in lines:
      pi_string += line.rstrip()

❸ print(pi_string)
  print(len(pi_string))
```

Спочатку програма відкриває файл і зберігає кожен рядок цифр в списку - точно так же, як це робилося в попередньому прикладі. У точці 1 створюється змінна pi_string для зберігання цифр числа "пі". Далі слід цикл, який додає до pi_string кожну серію цифр, з якої видаляється символ нового рядка 2. У точці 3 програма виводить рядок і її довжину:

3.1415926535 8979323846 2643383279

36

Змінна pi_string містить пропуски, які були присутні на початкукожного рядка цифр. Щоб видалити їх, досить використовувати strip () замість rstrip ():

filename = 'pi_30_digits.txt'

with open (filename) as file_object:

```
lines = file_object.readlines ()

pi_string = ''
for line in lines:
 pi_string + = line.strip ()

print (pi_string)
print (len (pi_string))
```

У підсумку ми отримуємо рядок, що містить значення «пі» з точністю до 30 знаків.

Довжина рядка дорівнює 32 символам, тому що в неї також включається початкова

цифра 3 і точка:

3.14159265358979323846264338327 9

32

Запис в файл

Один з найпростіших способів збереження даних - запис в файл. текст, записаний в файл, залишиться доступним і після закриття терміналу з висновком вашої програми. Ви зможете проаналізувати результати післязавершення програми або передати свої файли іншим. Ви також зможетенаписати програми, які знову читають збережений текст в пам'ять і працюють з ним.

Запис в порожній файл

Щоб записати текст в файл, необхідно викликати open () з другим аргументом, який повідомляє Python, що ви збираєтеся записувати дані в файл. Щоб побачити, як це робиться, напишемо просте повідомлення і збережемо його у файлі (Замість того щоб просто вивести на екран):

write_message.py

```
filename = 'programming.txt'
```

```
❶ with open(filename, 'w') as file_object:
❷     file_object.write("I love programming.")
```

При виклику open () в цьому прикладі передаються два аргументи 1. Перший аргумент, як і раніше, містить ім'я файлу, що відкривається. Другий аргумент 'w' повідомляє Python, що файл повинен бути відкритий в режимі запису. Файли можна відкривати в режимі читання ('r'), записи ('w'), приєднання ('a') або в режимі, допускає як читання, так і запис в файл ('r +'). Якщо аргумент режиму не вказано, Python за замовчуванням відкриває файл в режимі тільки для читання.

Якщо файл, що відкривається для запису, ще не існує, функція open () автоматично створює його. Будьте уважні, відкриваючи файл в режимі запису ('w'): якщо файл існує, то Python знищить його дані перед поверненням об'єкта файлу.

У точці 2 метод write () використовується з об'єктом файлу для запису рядка в файл. Програма не виводить дані на термінал, але, відкривши файл programming .txt, ви побачите в ньому один рядок:

programming.txt

I love programming.

Цей файл нічим не відрізняється від будь-якого іншого текстового файлу на вашому комп'ютері. Його можна відкрити, записати в нього новий текст, скопіювати / вставити текст і т. д.

Багаторядкова запис

Функція write () не встановлює символи нового рядка до записаного текст. А це означає, що якщо ви записуєте відразу кілька рядків без включення символів нового рядка, отриманий файл може виглядати не так, як ви розраховували:

```
filename = 'programming.txt'
with open (filename, 'w') as file_object:
file_object.write ( "I love programming.")
 file_object.write ( "I love creating new games.")
```

Відкривши файл programming .txt, ви побачите, що два рядки «склеїли»:

I love programming.I love creating new games.

Якщо включити символи нового рядка в команди write (), текст буде складатися з двох рядків:

filename = 'programming.txt'

with open (filename, 'w') as file_object:

file_object.write ("I love programming. \ n")

file_object.write ("I love creating new games. \ n")

Результат виглядає так:

I love programming.

I love creating new games.

Для форматування виводу також можна використовувати прогалини, символи табуляції і порожні рядки по аналогії з тим, як це робилося з висновком на термінал.

Приєднання даних до файлу

Якщо ви хочете додати в файл нові дані замість того, щоб перезаписувати існуюче вміст, відкрийте файл в режимі приєднання. В цьому випадку Python не знищує вміст файлу перед поверненням об'єкта файлу. Всі рядки, що виводяться в файл, будуть додані в кінець файлу. Якщо файл ще не існує, то Python автоматично створить порожній файл. Змінимо програму write_message .py і доповнимо існуючий файл programming .txt новими даними:

write_message.py

filename = 'programming.txt'

```
❶ with open(filename, 'a') as file_object:
❷     file_object.write("I also love finding meaning in large datasets.\n")
      file_object.write("I love creating apps that can run in a browser.\n")
```

У точці 1 аргумент 'a' використовується для відкриття файлу в режимі приєднання (Замість перезапису існуючого файлу). У точці 2 записуються дві нові рядки, які додаються до вмісту programming .txt:

programming.txt

I love programming.

I love creating new games.

I also love finding meaning in large datasets.

I love creating apps that can run in a browser.

В результате к исходному содержимому файла добавляется новый текст.

Давайте підведемо невеликий підсумок:

Python - це повноцінний і багатофункціональний мова програмування

Ця мова застосовується в самих різних сферах, в тому числі в web-програмуванні

Це ідеальна мову для того, щоб почати своє становлення як програміста, тому що він зрозумілий і простий у використанні

Є улюбленим мовою безлічі професійних програмістів

РОЗДІЛ 2. ВИВЧЕННЯ ЕФЕКТИВНОСТІ ТЕСТОВОГО КОНТРОЛЮ ЯК МЕТОДУ ОБ'ЄКТИВНОЇ ОЦІНКИ ЗНАНЬ, УМІНЬ І НАВИЧОК

2.1 Контроль як метод навчання, його функції

Контроль як педагогічне поняття являє собою усвідомлене, планомірне спостереження та фіксацію вербальних і практичних дій вихованців з метою з'ясування рівня набуття ними соціального досвіду, опанування програмного матеріалу, оволодіння теоретичними і практичними знаннями, навичками й уміннями та формування в них певних особистісних і професійних рис.

Отже, головна мета контролю як дидактичного засобу управління навчанням — забезпечення його ефективності приведенням до системи знань, умінь, навичок учнів, самостійного застосування здобутих знань на практиці, стимулювання навчальної діяльності учнів, формування у них прагнення до самоосвіти та контролю, наприклад у навчанні, полягає у навчанні, полягає у з'ясуванні рівня засвоєння програмного матеріалу, визначенні дієвості та ефективності організації навчального процесу, в оцінці якості викладання навчальних дисциплін. Контроль чи перевірку результатів навчання трактують у сучасній дидактиці як педагогічну діагностику.

Контроль знань учнів складається з: перевірки — виявлення рівня знань, умінь та навичок; оцінки — вимірювання рівня знань, умінь і навичок; обліку — фіксування результатів у вигляді оцінок у класному журналі, щоденнику учня, відомостях.

За допомогою контролю в процесі навчання розв'язують низку завдань: виявлення готовності учнів до сприйняття, усвідомлення і засвоєння нових знань; отримання інформації про характер самостійної роботи у процесі навчання; визначення ефективності організаційних форм, методів і засобів навчання; виявлення ступеня правильності, обсягу і глибини засвоєних учнями знань, умінь та навичок. Ці та інші завдання визначають зміст контролю, який змінюється із зміною дидактичних завдань.

«Педагогічна діагностика» досліджує навчальний процес, в ході якого вивчаються передумови, умови і результати навчального процесу, з метою оптимізації чи обґрунтування значення його результатів для суспільства, тобто з'ясовується рівень набуття знань суб'єктами учіння; формування у них практичних навичок, вмінь та їх міцність; рівень їхнього загального розвитку і вихованості; опрацювання й аналіз отриманих результатів.

Контроль знань у педагогічній літературі називають своєрідним методом навчання, що виконує різноманітні функції, найбільш важливими серед них є:

Система аналізу й оцінки знань, умінь та навичок учнів передбачає виконання таких основних функцій: навчальної, стимулюючої, виховної, діагностичної (рис. 2).

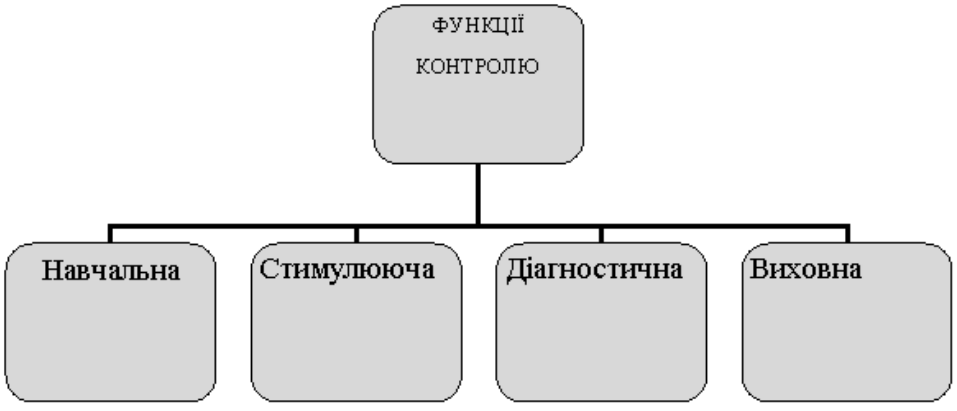

Рис. 2 Функції аналізу й оцінки

1. Навчальна функція полягає, у забезпеченні зворотного зв'язку як передумови підтримання дієвості й ефективності процесу навчання. У ньому беруть участь два суб'єкти: учитель й учні. Тому система навчання може функціонувати ефективно лише за умов дії прямого і зворотного зв'язків. У процесі навчання добре проглядається, в основному, прямий зв'язок (учитель знає, який обсяг знань має сприйняти й усвідомити учень), а складно, епізодично налагоджується зворотний зв'язок (який обсяг знань, умінь і навичок і як засвоїв кожний учень) (рис. 3).

2. Виховна, що полягає у формуванні вміння відповідально й зосереджено працювати, застосовувати прийоми контролю й самоконтролю, сприяє розвитку працелюбності, активності, формує в учнів відповідальність та інших позитивних якостей особистості.

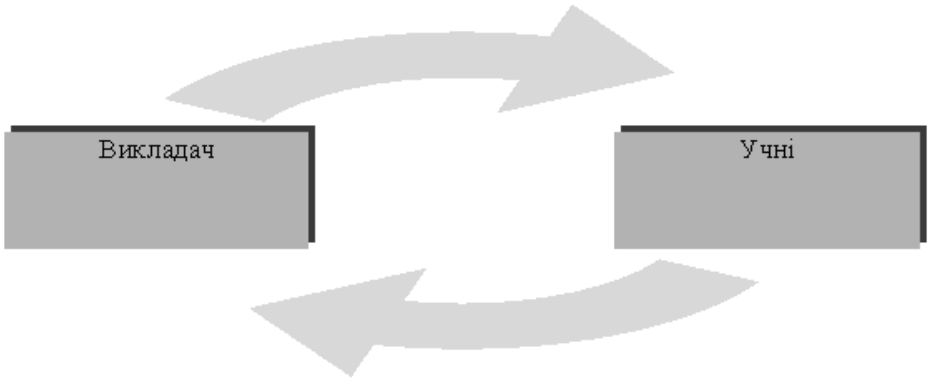

Рис. 3. Зв'язки між викладачем і учнями

3. Діагностична функція аналізу й оцінки знань, умінь та навичок передбачає виявлення прогалин у знаннях учнів, процес учіння має форму концентричної спіралі. Якщо на нижчих рівнях учіння трапилися прогалини, то буде порушена закономірність спіралеподібної структури учіння. Тому так важливо своєчасно виявити прогалини, усунути їх і лише потім рухатися вперед.

4. Стимулююча функція аналізу й оцінки навчальної діяльності учнів зумовлюється психологічними особливостями людини, що проявляється у

бажанні кожної особистості отримати оцінку результатів певної діяльності, зокрема навчальної. Це викликано ще й тим, що у процесі навчання студенти і школярі щоразу пізнають нові явища і процеси. Через недостатній рівень соціального розвитку та самооцінки учням важко об'єктивно оцінити рівень і якість оволодіння знаннями, вміннями та навичками. Учитель своїми діями і має допомогти учням усвідомити якість і результативність навчальної праці, що психологічно стимулюватиме школярів до активної пізнавальної діяльності.

5. Контролююча (учитель отримує зворотній зв'язок, визначає, як узагалі засвоюється навчальний матеріал,чи оптимально підібрані форми, методи й прийоми навчання).

6. Розвиваюча — в процесі навчання в учнів розвивається логічне мислення, зокрема вміння аналізу і синтезу, порівняння І узагальнення, абстрагування і конкретизації, класифікації та систематизації, розумова діяльність, мовлення, пам’ять, уява, увага;

7. Оцінювальна — об’єктивна оцінка знань, умінь і навичок учнів сприяє кращому навчанню;

8. Управлінська — на основі контролю визначається стан успішності учнів, що дає змогу запобігти неуспішності або подолати її. В цьому разі вчитель коригує і свою власну діяльність — змінює методику викладання, вдосконалює навчальну діяльність учнів. Реалізація цих функцій залежить від дотримання основних принципів перевірки навчально-пізнавальної діяльності та оцінки знань, навичок і вмінь учнів.

2.2 Види контролю

Контроль знань здійснюється на всіх етапах навчання. Його види класифікують за різними критеріями залежно від: способу здобуття інформації в процесі контролю; засобів, які використовують під час контролю і самоконтролю; способу організації контролю і форми організації контролю; дидактичної мети і місця застосування в навчальному процесі.

Залежно від засобів контролю його класифікують на машинний і безмашинний.

Машинний контроль найчастіше використовують у програмованому навчанні для управління процесом засвоєння знань. До його специфічних засобів належать тренажери, контролювальні та навчальні машини, комп'ютери. Машинний контроль дає змогу самостійно перевіряти результати навчальної діяльності, виявляти прогалини в знаннях, уміннях і навичках. Під час машинного Контролю відповіді кодуються і вводяться у відповідний пристрій, де машина порівнює їх з еталоном, після чого оцінює результати роботи.

Безмашинний контроль і самоконтроль здійснюють за допомогою програмованих підручників, усних і письмових завдань без використання будь-яких технічних приладів. У процесі безмашинного контролю відповіді учнів теж кодуються, але їх перевіряє вчитель за допомогою різних шаблонів.

Відповідно до способу організації контроль поділяють на програмований і непрограмований.

Суть програмованого контролю полягає у пред'явленні всім учням однакових стандартних вимог. Він забезпечує контроль за кожним кроком просування учня на шляху пізнання і дає змогу швидко діагностувати рівень засвоєння знань у великої кількості школярів, У програмованому контролі часто використовують тестові завдання, які забезпечують оперативний зворотний зв'язок у системі "навчальний матеріал - учень", ефективне самоуправління і самоконтроль навчальної діяльності.

Непрограмований контроль передбачає використання різних видів і форм діагностики результатів навчання відповідно до дидактичних вимог і правил без чіткого співвіднесення з конкретними формами організації і місцем застосування у навчальному процесі. За формами організації контроль може бути індивідуальним, фронтальним, груповим чи комбінованим.

Індивідуальне опитування спрямоване на перевірку знань, умінь і навичок окремих учнів. Поряд з можливістю ґрунтовної і всебічної перевірки особистісних досягнень школяра цей вид контролю має істотний недолік:

пасивність інших членів учнівського колективу у процесі контролю. Фронтальне опитування забезпечує можливість учителю за відносно стислий проміжок часу перевірити знання великої кількості учнів класу, спонукаючи їх до активності. Воно сприяє узагальненню і систематизації вивченого матеріалу.

Групове опитування спрямоване на перевірку знань, умінь і навичок групи учнів. У практиці сучасної школи часто використовують комбінований, або ущільнений, контроль - поєднання різних форм і методів організації контролю, наприклад коли один учень відповідає біля дошки, а решта виконує індивідуальні чи групові самостійні завдання.

За дидактичною метою і місцем застосування у навчальному процесі контроль класифікують на попередній, поточний, тематичний, періодичний, підсумковий, самоконтроль.

Основною метою попереднього контролю є діагностування, визначення загальної підготовленості учнів, рівня володіння основними поняттями предмета, вивчення якого вони розпочинають. Його іноді використовують перед вивченням нової теми чи на початку навчального року, семестру. Результати попереднього контролю стають основою для планування роботи на майбутнє, адаптації учнів до особливостей навчального процесу. Для його організації використовують різні види і методи контролю.

Поточний контроль спрямований на оперативне виявлення якості засвоєння учнями знань, навичок і вмінь на всіх етапах процесу навчання.

Його використовують учителі у повсякденній навчальній роботі і здійснюють зазвичай у формі систематичних спостережень, усного опитування, письмових робіт, фронтальної бесіди для оперативного оцінювання знань учнів і якості навчально-виховної роботи на уроці.

Тематичний контроль полягає у перевірці та оцінюванні знань учнів з кожної логічно завершеної частини навчального матеріалу (теми або розділу). Його використання передбачає попереднє ознайомлення учнів із загальною кількістю занять з теми; кількістю і тематикою основних робіт, термінами їх виконання; питаннями, що виносяться на атестацію; формою проведення

атестації та умовами оцінювання. Тематичну оцінку можна виставляти автоматично на основі поточних результатів навчання. Якщо тема велика за обсягом, то іноді проводять проміжні тематичні перевірки; якщо невелика - тематичний контроль доцільно здійснювати після вивчення декількох тем.

Періодичний контроль має на меті виявити, наскільки успішно учні оволодівають системою знань, який загальний рівень їх засвоєння щодо сучасних вимог. До нього належать внутрішньошкільна (директорська) та інспекторська перевірки. Внутрішньо-шкільну перевірку здійснює керівництво школи з метою контролю за якістю знань учнів і роботою вчителів, виконанням вимог навчальної програми, станом викладання предметів. Окрім цього вона передбачає виявлення нового, передового в практиці роботи вчителів, а також тих труднощів, з якими вони стикаються в процесі роботи, обґрунтування шляхів удосконалення педагогічної діяльності.

Внутрішньошкільна перевірка зазвичай відбувається на початку і наприкінці навчального семестру чи року. її результати обговорюють на педагогічній раді навчального закладу або узагальнюють у наказах директора. Інспекторська перевірка, яку здійснюють представники органів освіти, буває двох видів: фронтальна (охоплює всі сторони роботи школи) і тематична (охоплює окремі аспекти навчання, наприклад стан викладання якого-небудь предмета). Основними завданнями інспекторської перевірки є всебічне вивчення стану освітнього процесу в школі і своєчасне внесення коректив у його організацію. Їх здійснюють методами анкетування, тестування, вивчення документації, співбесіди та ін. За результатами перевірки складають підсумковий документ, який відображає джерела одержання інформації, зміст основних питань перевірки, висновки і пропозиції. Результати інспекторської перевірки ураховують під час атестації педагогічних працівників.

Основною метою підсумкового контролю є встановлення системи і структури знань учнів. Ця перевірка здійснюється наприкінці семестру (навчального року) або після завершення вивчення предмета. Основними формами підсумкового контролю є заліки та іспити.

Самоконтроль як важливий компонент свідомої саморегуляції поведінки і діяльності виявляється в усвідомленні власних особистісних станів, дій, знань, в узгодженні їх зі встановленими вимогами, правилами чи нормами. Мета самоконтролю полягає в запобіганні помилок та їх виправленні. Завдання вчителя спонукати учнів до об'єктивного оцінювання результатів навчання.

2.3 Оцінювання результатів навчально-пізнавальної діяльності учнів

Для ефективного контролю успішності учнів важливо не тільки виявити те, що вони знають і вміють, а й об'єктивно оцінити їхні знання та вміння. Підсумком контролю має бути оцінювання, яке передбачає зіставлення того, що школярі засвоюють, з тим, що вони повинні засвоїти відповідно до вимог навчальної програми.

Основними компонентами оцінювання є встановлення фактичного рівня знань, співвідношення виявлених знань з еталонними, оформлення результату навчальної діяльності учнів у вигляді оцінки-балів.

Оцінка - кількісний показник якості результатів навчально-пізнавальної діяльності учнів.

Єдині вимоги до оцінювання знань, умінь та навичок формулюються у вигляді критеріїв і норм. Критерій - міра оцінки, показник, на основі якого визначається рівень оволодіння знаннями, уміннями і навичками. Відповідно до критеріїв визначають норми оцінок - конкретні вимоги, які регулюють виставлення оцінок-балів з навчального предмета за усну відповідь чи письмову роботу. Наприклад, якщо основним критерієм оцінки письмової роботи є точність виконання, яка характеризується кількістю помилок, то норма для оцінки 10 балів - повна відсутність помилок. Норми відображають найтиповіші випадки і ситуації під час перевірки й оцінювання знань. Вони визначені у навчальних програмах з усіх предметів.

Основними критеріями оцінювання знань є:

1) глибина - кількість усвідомлених учнем істотних зв'язків і відношень у знаннях;

2) повнота - кількість усіх елементів знання про вивчений об'єкт;

3) міцність - збереження в пам'яті вивченого матеріалу, безпомилковість його відтворення;

4) оперативність - уміння учня використовувати знання у стандартних однотипних умовах;

5) якість - критерій, що охоплює повноту, міцність, глибину, оперативність знань тощо;

6) гнучкість - уміння учня використовувати знання у змінних, варіативних умовах;

7) систематичність - засвоєння навчального матеріалу в його логічній послідовності та наступності.

За цими критеріями визначають рівні знань учнів: репродуктивний (знання сприйняті, зафіксовані в пам'яті і можуть бути відтворені); реконструктивний (знання застосовуються в стандартних або варіативних умовах); творчий (знання продуктивно застосовуються в змінених, нестандартних ситуаціях).

З метою забезпечення ефективних вимірників якості навчальних досягнень та об'єктивного їх оцінювання запроваджено 12-бальну шкалу оцінювання, у якій ураховується рівень особистих досягнень учня. Критерії оцінювання при цьому не поділяють на позитивні та негативні. Серед загальних критеріїв оцінювання навчальних досягнень учнів у системі загальної середньої освіти виокремлюють:

- характеристику відповіді учня (правильність, цілісність, повнота, логічність, обґрунтованість);

- якість знань (осмисленість, глибина, гнучкість, дієвість, системність, узагальненість, міцність);

- сформованість загальнонавчальних і предметних умінь та навичок;

- рівень володіння розумовими операціями (аналіз, синтез, порівняння, класифікація, узагальнення тощо);

- розвиток творчих умінь (уміння виявляти проблему, формулювати гіпотезу, перевіряти її);

- самостійність оцінних суджень.

На основі цих критеріїв розрізняють рівні навчальних досягнень учнів:

1) початковий (рецептивно-репродуктивний), що характеризується первинними уявленнями про предмет вивчення, фрагментарністю відповідей учнів;

2) середній (репродуктивний), опанувавши який учень здатний розв'язувати найпростіші завдання за зразком, відтворювати основний зміст навчального матеріалу, володіє елементарними навчальними вміннями;

3) достатній (конструктивно-варіативний) характеризується знанням суттєвих ознак понять, оперуванням ними, розв'язуванням стандартних завдань, умінням робити висновки, виправляти допущені помилки, однак невмінням переносити і використовувати знання в інших навчальних ситуаціях;

4) високий (творчий), ознаками якого є систематизоване застосування їх для виконання творчих завдань, самостійне оцінювання різних явищ, фактів, уміння обстоювати особисту позицію.

Контроль за навчальною діяльністю здійснюється різними методами і в різних формах. їх використання спрямоване на виявлення й оцінювання знань, а також фіксацію результатів навчання. У шкільній практиці застосовують такі методи контролю: спостереження за різними видами діяльності учнів; усне опитування; письмову перевірку (контрольна робота, твір, письмове домашнє завдання та ін.); графічну перевірку; перевірку практикою (лабораторні та практичні роботи); тести успішності; самоконтроль.

РОЗДІЛ 3. ВИКОРИСТАННЯ ЕОМ ДЛЯ КОНТРОЛЮ

3.1 Особливості і обмеження комп’ютерного контролю знань

Відомо, що у навчального контролю багато педагогічних функцій. При розгляді комп’ютерних засобів контролю мається на увазі тільки одна з них, а саме, - контролююча. Метою контролю в даному випадку є встановлення рівня знань контролюючої особи, в спрощеному розумінні - виставляння оцінки за знання. Більш широку оцінку знань за допомогою комп’ютерного контролю важко здійснити в зв’язку зі специфічними обмеженнями, які накладаються комп’ютером на можливість створення відповіді на поставлене питання.

Створення відповіді на поставлене в завданні при проведенні контролю питання в звичайних системах, тобто в системах без аналізу особливостей інтелектуальних програм контролю, неможливий шляхом аналізу семантики відповіді, виконаного в довільній мовній формі. Звичайні комп’ютерні програми контролю можуть тільки співставляти введену відповідь з кодом, який характеризує правильну відповідь. Результатом такого співставлення є фіксація збігу чи неспівпадань. В більшості програм збігу приписують одиницю (відповідь правильна) чи нуль (відповідь неправильна), коли збігу немає. Звісно, можна також зафіксувати відмови від відповіді. Але ця відмова є неінформативною і, як правило, не використовується.

Основні типи відповідей, які досить просто можна реалізувати в рамках перевірки кодованої відповіді:

Виборочна відповідь. Питання (завдання) формулюється так, що на нього можна привести набір варіантів відповідей, кожен з яких позначається кодом (цифрою, символом, набором символів, картинкою і т.п.). Серед пропонованих варіантів відповідей може бути один правильний, хоча б один вірний, декілька вірних, прочому потрібно вказати або всі вірні, або ж їх потрібну кількість. Інколи форму вибору відповіді ускладнюють, пропонуючи створити деякий набір з декількох груп, в кожній з яких потрібно вибрати ту чи іншу

компоненту з представленого в кожній групі набору. Програма зіставляє введений код з кодом, який розміщений в пам’яті програми комп’ютера і фіксується відповідь в бінарній системі.

Числова відповідь. Потрібно розв’язати задачу або виконати деякі дії, в результаті яких одержується число. Комп’ютер виконує перевірку введеного числа зх. Числом в пам’яті комп’ютера. При цьому у відповіді утримується потрібна кількість знаків.

Перевірка простої формули. Відповідь потрібно ввести у вигляді не дуже складної формули, правильність якої можна перевірити простим способом, наприклад співставленням результатів обчислень по введеній і правильній формулам. У введену і правильну формулі програмою перевірки звичайно підставляються випадкові числа.

Перевірка логічної формули. У відповідь на поставлене питання вводиться деяка послідовність слів або виразів, перевірку наявності чи відсутності яких можна виконати за допомогою раніше введеної в програму логічної формули.

Перевірка слова, послідовності слів чи інших символів, введених у відповідь, сформульований у вигляді відкритого питання з пропуском цих слів, які потрібно ввести.

Перераховані можливості сильно обмежують дидактичні можливості перевірки правильності відповіді і впливають на вибір тих видів навчальної діяльності, в яких можна використовувати комп’ютерну форму перевірки. Дійсно, такого роду перевірка допустима (і широко використовується) при оперативному чи поточному контролі, можливо – при кінцевому контролі, а також використовується для попереднього в більш значимих видах навчального контролю, наприклад, підсумковому. Очевидно, що ці обмеження в значній мірі зменшують впевненість в тому, що результати контролю адекватно відображають дійсні знання контролюючих, що дуже важливо при випускному контролю чи при вступних екзаменах у вуз. Саме відмінності в психічній діяльності індивідуума при розумінні ним відповіді на поставлене питання чи при виборі потрібної відповіді з множини представлених варіантів, не

дозволяють зробити висновки про характерні особливості розумової діяльності і ступеня сформованості знань по навчальній дисципліні. Недарма багато керівників провідних вузів країни проти такого способу відбору абітурієнтів у вузи.

3.2 Підходи до зіставлення комп'ютерних програм контролю

Розглянемо особливості підходів до створення наборів контрольних завдань для організації комп'ютерного контролю знань. Такі набори називаються тестами (предметно-орієнтованими тестами), хоча між наборами завдань і тестом маються значні відмінності. Питання про те, при виконанні яких вимог набір можна вважати тестом, широко обговорювалося в журналі «Питання тестування в навчанні», де йшлося про те що, набір завдань, який претендує на назву тесту, повинен, по крайній мірі, задовольняти вимогам валідності та надійності.

Існує декілька підходів до створення наборів тестових завдань, які можуть бути заложені в комп'ютерну програму. Розглянемо два з них.

ТРАДИЦІЙНИЙ ПІДХІД

Цей підхід детально описаний в роботах В.С.Аванесова і книзі Г.А. Атанова.

При реєстрації результатів виконання кожного з наборів завдань в дихотомічній системі (успіх – одиниця, неуспіх – нуль) принципіальним є питання про однакову складність пропонованих завдань, тобто про складання набору однакової складності завдань. Тільки такий набір може правильно відображати результати перевірки. З точки зору статистики можна уявити, що мається однорідна по складності гіпотетична генеральна сукупність, з якої проведена неповоротна вибірка, яка складає даний набір завдань. Оскільки ніякої апріорної впевненості в створеному таким чином наборі завдань про однорідність немає, потрібно здійснити деяку селекцію завдань, виключивши з набору ті, які мають «випавши з ряду» складність.

За результатами перевірки складається таблиця, в якій наявний стовпчик списку учасників, стовпці з переліком номерів вправ і результатами

випробувань, тобто нулями і одиницями розставленими по рядкам списку. Для зручності таблицю доповнюють рядком сумарних відміток по стовпцям (по кожній вправі) і стовпцем сумарних результатів по кожному учню. Далі над таблицею виконується дія «чистки»: з неї видаляються вправи, «випадаючі» за складністю із ряду інших. Спершу проводиться просте співставлення результатів для всіх вправ. Якщо для якої-небудь вправи відповіді майже всіх учнів були рівні одиниці, тобто результат явно перевищує результати випробувань для решти вправ, отже ця вправа явно легша за інші, володіє малою селективністю і повинна бути видалена зі списку. Аналогічно, якщо сумарний результат випробувань деякої вправи близький до нуля (має мале значення), вправа дуже складна. Вона також не селективна, і її також потрібно видалити. Після цього первинного етапу виконується подальша «чистка» таблиці.

Порівнюючи результати, отримані для кожної вправи (ряд нулів і одиниць кожного стовпця) з числами останнього стовпця, де записані сумарні дані. Ті вправи, які не корелюють з даними останнього стовпця, також потрібно видалити. Мірою порівняння являється коефіцієнт кореляції Пірсона, який записується в додатковому нижньому рядку, створеному спеціально для цього, для кожної вправи. Відкидаються ті завдання, коефіцієнт кореляції для яких менше по абсолютному значенні 0,3 (умовна норма). На цьому етапі чистка не завершується. Наступний етап складається з визначення коефіцієнтів кореляції Пірсона між стовпцями таблиці: кожної вправи з кожною з усіх останніх, тобто першого стовпця з самим собою (це 1), з другим, третім і т.д. Другого з першим, другим і т.д. По цим коефіцієнтам створюється кореляційна матриця. Вона квадратна, симетрична, з одиницями по головній діагоналі.

Далі визначаються середні коефіцієнти кореляції для кожної з вправ (стовпців), які порівнюються між собою. З матриці видаляються ті стовпці, для яких середній коефіцієнт кореляції «випадає» з ряду, тобто значення яких не корелюється з іншими і їх коефіцієнт кореляції менше 0,3.

В результаті отримується нова таблиця, по якій будується нова кореляційна матриця, - і так далі, до певної межі. Отримана в результаті таблиця претендує на назву теста. Для цього перевіряють, наскільки корелюють між собою сумарні результати, отримані по різним (рівним за розмірами) частинами «очищеної» таблиці, наприклад парним і непарним стовпцями чи лівими і правими половинами таблиці. Так складаються результати випробувань для вправ з непарними номерами (один стовпець) і парними номерами вправ (другий стовпець) і визначається коефіцієнт кореляції Пірсона між цими стовпцями. При значенні коефіцієнту кореляції вище 0,8 (умовна норма) набір завдань можна вважати такими, що задовольняють вимоги надійності.

Описана процедура дозволяє створити набір приблизно рівно складних завдань.

Модель з урахуванням неоднорідності складності завдань в наборі

Суть цієї моделі полягає в тому, що результати виконання завдання враховується не нулями і одиницями, а нарахуванням і відніманням балів, рівних (чи пропорційних) апріорі назначеної складності завдань з наступним обрахунком середнього набраного балу. Результат усереднення переводиться (перераховується) в оцінку в будь-якій вибраній шкалі оцінок.

Для пояснення способу припустимо, що маємо базу завдань, яка охоплює перевірку знань в рамках теми. Кожному із завдань приписується та чи інша складність в прийнятій шкалі, наприклад 0-100 балів, розбитій для простоти рівномірно на 10 частин: 1-10, 10-20 і т.д. Нехай припустимо спочатку, що завдання розподілені рівномірно на шкалі складності, симетрично відносно центрального значення X (тут X=50 балів) і заповнюють шкалу протяжністю 2A, а врахування складності проводиться з точністю до 10. Рівномірно означає, що кількість вправ в рамках кожної ділянки майже однакова. За кожне правильно виконане завдання нараховується число балів, яке рівне складності завдання. За неправильне – скидається число, доповнюючи складність до 100 балів. Таким способом враховується положення: помилка штрафується тим

вище, чим легше завдання, і навпаки, чим менше, тим завдання складніше. На рисунку 1 показані дві потовщені опорні лінії, які перетинаються, одна з яких відповідає правильним відповідям, інша – неправильним. Для розглядуваного випадку рівняння прямих такі:

для правильних відповідей – y=x

для неправильних – y=-x+2x

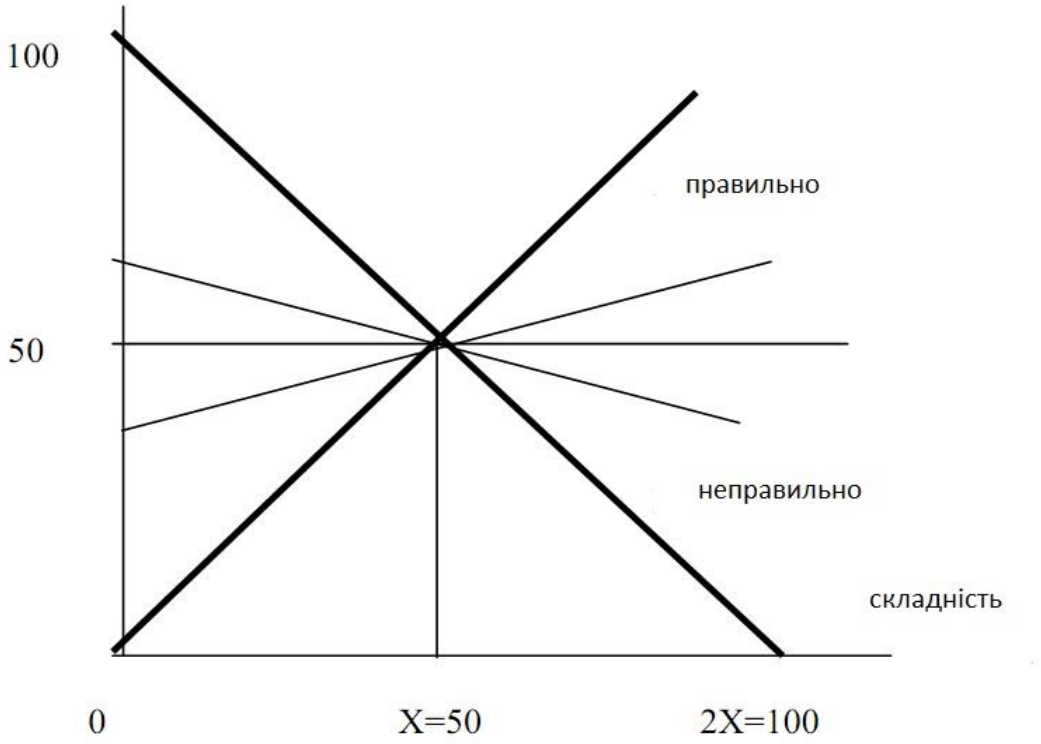

Рис. 1

Завдання пропонуються особі, яка проходить контроль, в наступному порядку.

На кожному непарному кроці на ділянці А, розташованою, наприклад, з права від значення X датчиком випадкових чисел завдання вибирається завдання з деякою випадковою складністю. На кожному парному кроці складність завдання вибирається автоматично симетрично (дзеркально) відносно значення X (рис.2), тобто у відповідності з рівняннями

$$x_{k-1} = 2X - x_k = 100 - x_k$$

де хк – складність непарного завдання (к = 1, 3, 5…). Використані завдання вибувають з числа пропонованих в поточному сеансі випробувань.

Після кожної пари виповнених завдань підраховується середнє значення накопичених балів, яке в кінці сеансу випробувань представляє собою відмітку в шкалі 100.

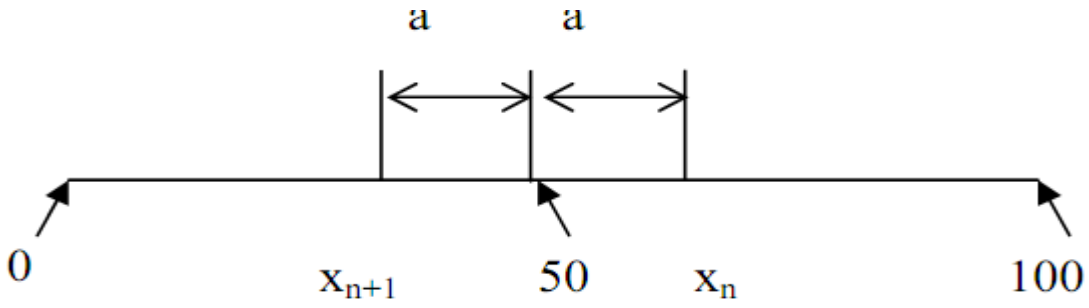

Рис. 2

При безпомилковому виконанні середнє число накопичених балів рівне X=50. Для виконуючих завдання, які здійснюють помилки, це значення завжди менше 50. Результат підрахованих таким чином багаторазових пропонованих пар перераховуються в залікову відмітку в будь-якій зручній шкалі, наприклад 100-бальній.

Прийняте в описі значення X=50 визначає середню складність набору завдань, симетрично розміщених на шкалі складності відносно цього значення. Зміщуючи це значення по шкалі складності, можна створювати набори різних середніх складностей.

Вимога щодо симетрії може бути замінена на більш просту, а саме, число завдань на парній стороні повинно бути не менше числа на непарній при рівномірному «заселенні» сторін. Це можна виконати шляхом додавання та видалення деяких завдань.

Описаний спосіб володіє відомою гнучкістю. Так, зменшуючи нахил опорних прямих (за абсолютним значенням), можна регулювати, так би мовити, «селективність» системі контролю (на рис.1 – тонкі опорні прямі). міняючи положення X на шкалі складностей («центра складності»), можна регулювати (установлювати) середню складність набору завдань. Міняючи нахил прямих – «селективні» властивості – послаблювати залежність нарахованих і віднятих

балів за складність завдання. Довжина ділянки 2А відповідає зміні діапазону занесення до бази даних завдань різної складності. Його можна зробити як вузьким, так і досить протяжним.

3.3. Створення тестових завдань

Для оцінювання успішності як учнів, так і студентів використовуються різноманітні типи тестових завдань, проте основним типом залишаються багатовибіркові завдання. Тестове завдання багатовибіркового типу складається з умови та запропонованого списку варіантів відповідей, з яких екзаменований повинен вибрати правильну чи найкращу відповідь (або кілька відповідей). Для створення ефективних тестових завдань необхідно дотримуватися певних принципів.

Тестове завдання багатовибіркового типу складається з двох частин: умови, яка описує певну проблему та ставить завдання перед екзаменованим, та запропонованого списку варіантів відповідей, серед яких як мінімум один є правильною чи найкращою відповіддю, а решта - дистрактори - є неправильними відповідями.

Тестові завдання багатовибіркового типу

Формат А (тестові завдання з однією найкращою відповіддю)

Тестові завдання (ТЗ) з вибором однієї найкращої відповіді є найбільш широко вживаними. Вони складаються з умови завдання (клінічна/ лабораторна ситуація), ввідного запитання та 4-5 варіантів відповіді, один з яких правильний.

Формат Х (тестові завдання множинного вибору правильно / неправильно)

За зовнішнім виглядом цей формат ідентичний формату А, проте правильними може бути будь-яка кількість відповідей.

При розгляді завдань формату А та Х постає проблема порівняння завдань, які ці формати ставлять перед екзаменованим. На думку S.Case & Swanson ці формати ставлять не ідентичні завдання. Відповідаючи на завдання формату Х,

екзаменовані повинні вирішувати, наскільки правильним є кожен запропонований варіант для того, щоб бути обраним як правильна відповідь. В той час, коли в завданнях формату А дистрактори не є абсолютно неправильними, в завданнях формату Х кожен варіант відповіді повинен бути або абсолютно правильним, або ж абсолютно неправильним, для того, щоб уникнути неоднозначності. Для виконання цієї умови автори тестових завдань, які складають ТЗ формату Х, найчастіше, на відміну від ТЗ формату А, перевіряють тільки засвоєння фактів та концепцій, тобто декларативних, а не процедурних знань.

Формат N (тестові завдання з декількома найкращими відповідями)

Як альтернативу формату Х (множинний вибір правильно/ неправильно) та негативно сформульованим тестовим завданням (Що з вказаного …, ОКРІМ?; Що з вказаного НЕ …?) декілька авторів пропонують використовувати формат N. У тестовому завданні формату N екзаменований повинен вибрати певну кількість (наприклад, 2, 3 чи 4) відповідей з запропонованого списку варіантів відповідей. Перелік варіантів відповідей, як правило, досить довгий (до 30 елементів). Формат N доцільно використовувати в таких ситуаціях, як, наприклад, збір анамнезу, дослідження, рекомендації, профілактика та невідкладні стани, де доречно вказувати більше однієї відповіді. Доцільність цього формату стане більш очевидною, коли ми перейдемо до розгляду форматів ключових моментів та логічних пар.

Формат ключових моментів

Для кращого оцінювання вміння приймати клінічні рішення Медична Рада Канади розробила формат тестових завдань, що базується на концепції ключових моментів. Ключовий момент визначається як критичний етап у вирішенні клінічної проблеми. Перелік клінічних проблем міститься у освітніх цілях, розроблених Медичною Радою Канади для кваліфікаційних іспитів. Кожна проблема описана у вигляді скарг пацієнта і супроводжується списком відповідних діагнозів. Освітні цілі, клінічні проблеми і відповідні діагнози складають опис домену знань та вмінь майбутнього лікаря. Додаткові

специфікації тестових завдань, такі як вік пацієнта, стать і клінічна ситуація, також беруться до уваги при створенні тестових завдань, проте і домен, і специфікації визначаються ключовими моментами проблем.

Формат R (тестові завдання розширеного вибору)

Цей формат відноситься до категорії логічних пар і ще називається форматом розширеного підбору. В цьому форматі кожен блок складається з 3-4 умов та спільного для них переліку варіантів відповідей. Кількість варіантів відповідей може коливатися від 4 до 20.

Ситуаційний кластер

Хоча кластер не вважається окремим форматом тестових завдань в чистому вигляді, проте він широко використовується в різних дисциплінах. Кластер складається з ввідного сценарію (опису ситуації) і наступними завданнями, що прямо стосуються цього сценарію.

До кожного сценарію ставиться два і більше запитань. Характер цих запитань визначається змістом та елементами компетентності, вказаним в домені. При створенні кластерів необхідно стежити, щоб запитання були пов’язані зі сценарієм та не “зчеплені” між собою. Це означає, що відповіді на запитання повинні бути незалежними одне від одного, тобто відповідь на одне запитання не повинна бути умовою для відповіді на інші запитання з цього ж кластера і не повинна впливати на них.

Формат B

Тестові завдання формату B відносяться до категорії логічних пар, що складаються з 3-5 визначень або цифрових значень, які є варіантами відповідей, та переліку слів чи фраз, які містять завдання. Варіанти відповідей означені буквами, а слова чи фрази, які містять завдання, - пронумеровані. Екзаменований повинен до кожного пронумерованого завдання підібрати один найбільш відповідний варіант відповіді, означений буквою.

Оскільки кожен варіант відповіді може використовуватися більше одного разу або не використовуватися взагалі, тестові завдання формату B неможливо вирішити методом виключення. Передбачалося, що тестові завдання цього

формату розширять використання завдань багатовибіркового типу, дозволяючи оцінити кілька взаємозв'язаних предметів в одному блоці. На відміну від сучасних форматів категорії логічних пар, завдання формату B часто не містили ввідного запитання, в результаті чого були досить неоднозначними. Загалом цей формат успішно використовувався до останнього часу, коли він був витіснений форматом R.

Формат D

Тестові завдання формату D є комплексними завданнями з категорії логічних пар. Кожне тестове завдання складається з трьох категорій, означених буквами, та п'яти пронумерованих ситуацій. Екзаменований повинен виконати два наступні завдання: 1) визначити категорію, з якою пов'язані 4 з 5 ситуацій і 2) визначити ситуацію, яка не відноситься до тієї ж категорії, що і решта чотири.

Передбачалося, що такі завдання нададуть можливість виявити розуміння відмінностей між рядом подібних факторів. Проте цей формат виявився складним для написання, а інструкції до його застосування – досить заплутаними. Окрім цього формат D має низьку розподільну здатність.

Формат K

Формат K був найбільш поширеним форматом з категорії правильно/ неправильно. Завдання цього формату складаються з умови та чотирьох варіантів відповідей, один чи більше з яких були правильними. Завдання екзаменованого полягало у виборі правильної комбінації варіантів відповідей.

Вважалось, що тестові завдання формату K повинні перевіряти глибокі знання та розуміння різних аспектів, процесів чи методів і вимагати від екзаменованого знання різних фактів за даною темою. Однак виявилось, що такі тестові завдання мають зайву складність, вимагають від екзаменованого запам'ятовування правильної комбінації та відповідної їй літери. Окрім того, можлива комбінація відповідей являє собою підказку, що значно знижує розподільну здатність тестового завдання та надійність тесту. Складно створити якісні однозначні завдання цього формату. Оскільки вони можуть

включати лише абсолютно правильні або абсолютно неправильні факти, то неможливо використовувати цей формат для оцінки клінічного мислення, окрім як в порівнянні (наприклад, “препарат X краще препарату Y для лікування захворювання Z”). Також ці завдання виявилися менш ефективними, ніж завдання інших форматів багатовибіркового типу, і мали меншу відносну достовірність на одиницю часу тестування.

Формат C

Тестові завдання формату C зовні схожі на завдання формату B, проте екзаменований повинен визначити правильні/ неправильні відповіді. Тестове завдання цього формату складається з переліку визначень, які є варіантами відповідей, та переліку слів чи фраз, які містять завдання. Варіанти відповідей означені буквами, а слова чи фрази, які містять завдання, - пронумеровані. Екзаменований повинен вирішити щодо кожного пронумерованого завдання, чи є правильною відповідь A, чи є правильною відповідь B, чи правильні обидві відповіді (варіант C), чи жодна з них неправильна (варіант D), причому кожен варіант відповіді може використовуватися більше одного разу або не використовуватися взагалі.

Цей тип тестових завдань використовувався для порівняння та протиставлення двох певних фактів. За рівнем складності завдання формату C відповідають завданням формату K. Основна проблема формату C полягає в тому, що екзаменований повинен визначити, наскільки “правильна” кожна відповідь, щоб бути вибраною як еталон відповіді. Наприклад, якщо завдання асоціюється з варіантами відповідей A і B, проте більше підходить до A, то екзаменований повинен вирішити: вибирати йому відповідь A чи обидві A і B. Якщо завдання слабо асоціюється з варіантами відповідей A і B, то екзаменований повинен визначити ступінь асоціації чи вибрати варіант D (всі відповіді неправильні). Такі типи суджень, що вимагаються від екзаменованого при відповіді на тестові завдання формату C, не пов’язані зі знаннями, а примушують екзаменованого здогадуватися про те, що саме автор мав на увазі.

Формат E

Тестові завдання формату E з множинними правильними чи неправильними відповідями базуються на аналізі взаємозв'язків. Завдання цього формату складається з речення з двома основними частинами: твердження і причина до цього твердження. Відповідаючи на такі завдання екзаменований повинен був вибрати варіант відповіді A, якщо обидві частини були істинними твердженнями і причина була правильним поясненням першого твердження; B – якщо обидві частини були істинними твердженнями, але причина не була правильним поясненням першого твердження; C – якщо перша частина була істинним твердженням, а друга – хибним; D – Якщо перша частина була хибним твердженням, а друга – істинним; E – якщо обидві частини були хибними твердженнями.

Вважалось, що для правильної відповіді на такі завдання необхідно критично мислити і розуміти основні принципи. Проте тестові завдання формату E складні для створення і заплутані для екзаменованих.

Формат H

Тестові завдання формату H є завданнями на кількісне порівняння. Ці завдання складаються із спарених тверджень, що описують два порівнюваних об'єкти, і варіантів відповідей: A – якщо A більше, ніж B, B – якщо B більше, ніж A і C – якщо обидва приблизно однакові.

Загальновизнано, що слід обмежувати тестові завдання, які вимагають запам'ятовування абсолютних кількісних величин, проте вважається, що тестові завдання формату H корисні в тих випадках, коли знання кількісної інформації є важливим. Проблема для екзаменованих у вирішенні таких завдань полягає у визначенні того, наскільки великою повинна бути різниця, щоб бути значимою.

Формат I

Тестові завдання формату I подібні на завдання формату H. Вони містять пари фраз, які описують певні параметри, що можуть змінюватися відносно один одного. Екзаменований повинен вибрати варіант відповіді A, якщо обидва параметри мають прямо пропорційний зв'язок (зростання параметра з першої

фрази супроводжується зростанням параметра з другої фрази), варіант відповіді В – якщо параметри мають обернено пропорційний зв’язок (зростання параметра з першої фрази супроводжується зменшенням параметра з другої фрази, і навпаки), варіант відповіді С – якщо параметри змінюються незалежно один від одного.

Ні тестові завдання формату Н, ні тестові завдання формату І не отримали широкого поширення. Через меншу кількість варіантів відповідей у порівнянні з іншими форматами тестових завдань вірогідність вгадування правильної відповіді в них дуже велика. Окрім цього, такі завдання концентруються на окремих деталях, а не на наукових положеннях.

РОЗДІЛ 4. СТВОРЕННЯ ПРОГРАМИ КОТРОЛЮ ЗНАНЬ ШКІЛЬНИХ ДИСЦИПЛІН ЗА ДОПОМОГОЮ CONSTRUCT CLASSIK

4.1. Construct classic

Construct classik - безкоштовний, заснований на DirectX9, ігровий конструктор для Windows, призначений для створення 2D ігор. Він використовує зручну і просту систему, засновану на подіях. За допомогою неї ви можете без праці створювати правила, за яким буде працювати ваша гра.

Construct зрозумілий для новачків і досить потужний для професіоналів. Якщо ви розчарувалися в складних логічних схемах в інших програмах - спробуйте Construct. Він простіше, ніж здається.

Construct не є комерційною програмою і розробляється добровольцями. Ви прямо зараз можете безкоштовно скачати повнофункціональну версію програми без жодних надокучливих вікон, реклами або обмежень. Повнофункціональна версія програми абсолютно безкоштовно.

Сам Construct поширюється по GPL ліцензії. Однак ця ліцензія не поширюється на будь-які твори, створені в програмі. Ви можете без всяких відрахувань або обмежень поширювати, продавати свої твори (Ігри, програми і

т.д.), створені за допомогою Construct, а також випускати їх з закритим вихідним кодом.

Дизайнери інді-ігор, художники або просто любителі можуть використовувати Construct для того, щоб швидко і легко створювати свої власні віртуальні світи без використання програмування. Вчителі та студенти можуть використовувати Construct, щоб вчитися принципам логіки в цікавій ігровій формі. Розробники можуть використовувати програму для швидкого створення макетів і прототипів, або просто як альтернативний, більш швидкий спосіб кодування. Ким би ви не були, Construct має безліч можливостей, щоб допомогти вам у вирішенні вашої задачі.

Інтерфейс програми Construct має безліч різних вкладок, які забезпечують швидкий доступ до широкого набору інструментів. інтерфейс повністю настроюється, тому ви можете реорганізувати його за вашим бажанням.

Він включає в себе панель управління проектом, вкладку шарів, вкладку анімації і панель властивостей(1.1 и 1.2). Ці панелі можна від'єднати і прикріпити в будь-якому зручному для вас місці, поміняти їм розмір і включити автоматичне приховування.

Редактор рівня повністю візуальний, побудований за принципом WYSIVYG, що робить процес побудови рівнів гри простим і захоплюючим. Ви можете в реальному часі переглядати ефекти, які ви додали, обертати і змінювати розмір об’єктів, змінювати налаштування проекту на панелі Properties (Властивості).

Також в програмі є вбудований редактор зображень, який дозволяє створювати спрайт, іконки і текстури для об’єктів на рівні. Об’єкти можна розташовувати на різних шарах, що дозволяє створити ефект паралакс і поліпшити організацію рівнів.

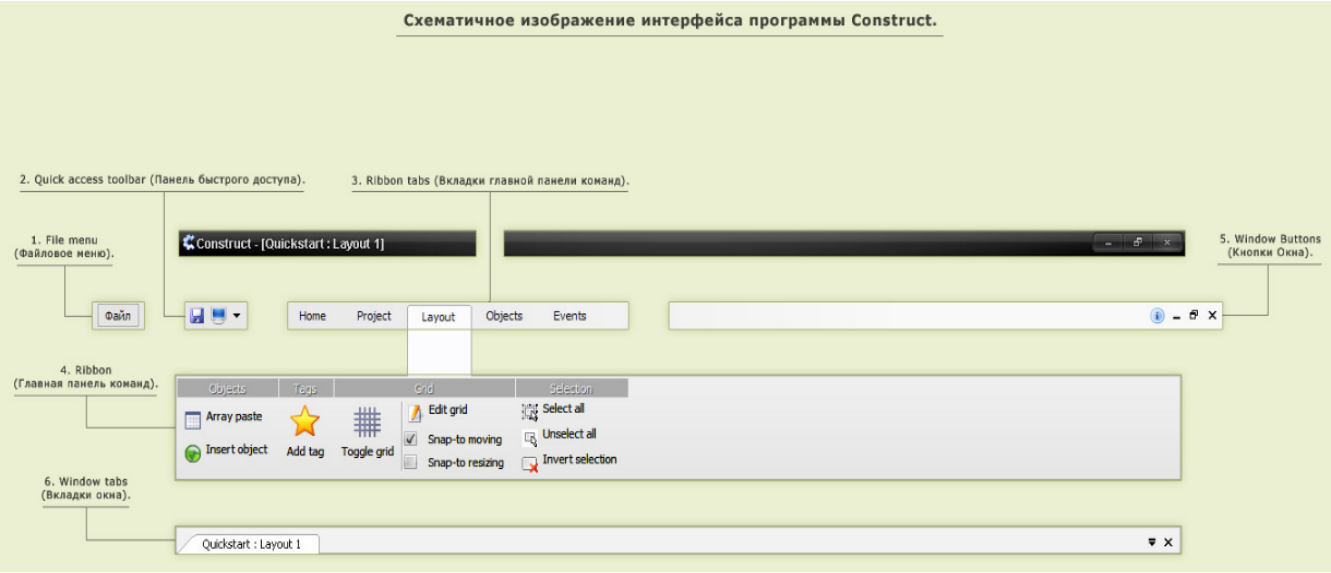

Рис 1.1

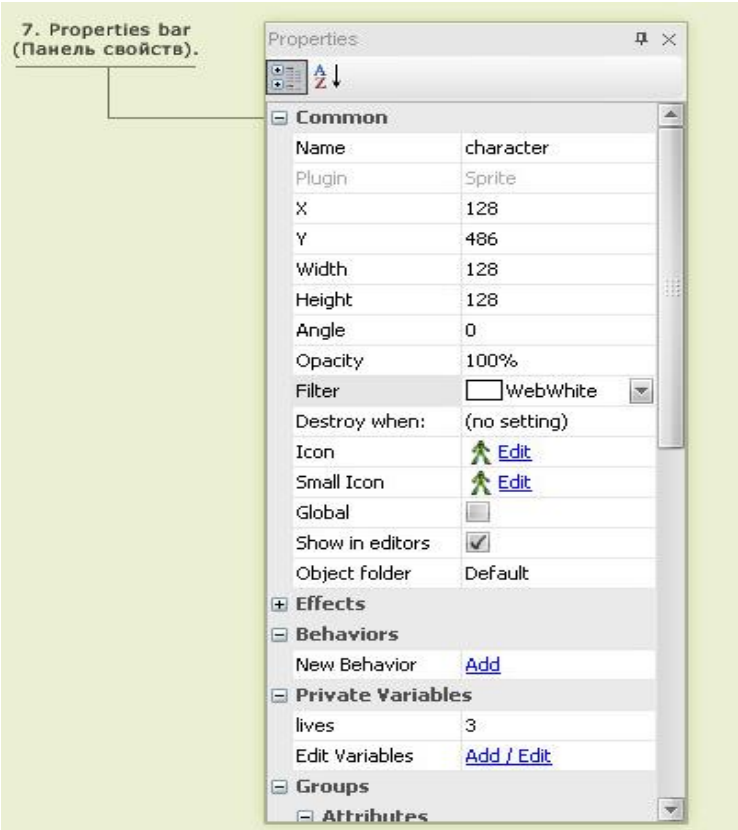

Рис. 1.2

Події в Construct представляють собою список логічних блоків, які складаються з умов і дій, які в свою чергу визначають, як саме повинна працювати ваша гра. Події в Construct є дуже потужним інструментом, який

здатний створювати абсолютно будь-які ігри без рядка програмування, що особливо добре підходить для новачків. Але навіть досвідчені програмісти можуть отримати результати набагато швидше в порівнянні з традиційним програмуванням.(1.3)

Події створюються шляхом вибору можливих умов. Це дуже просто. наприклад, ви створюєте умова - коли мій персонаж торкнувся золотої монети. Тепер ви вибираєте дію на цю умову, наприклад, взяти монету і додати вашому персонажу 1000 очок. Список подій створений максимально зрозумілим і інтуїтивним, щоб ви без зусиль могли отримати миттєвий результат. Цілі списки подій можуть бути легко згруповані, скопійовані, перенесені і навіть використовуватися на інших рівнях, що позбавляє вас від постійного відтворення подій для кожного рівня.

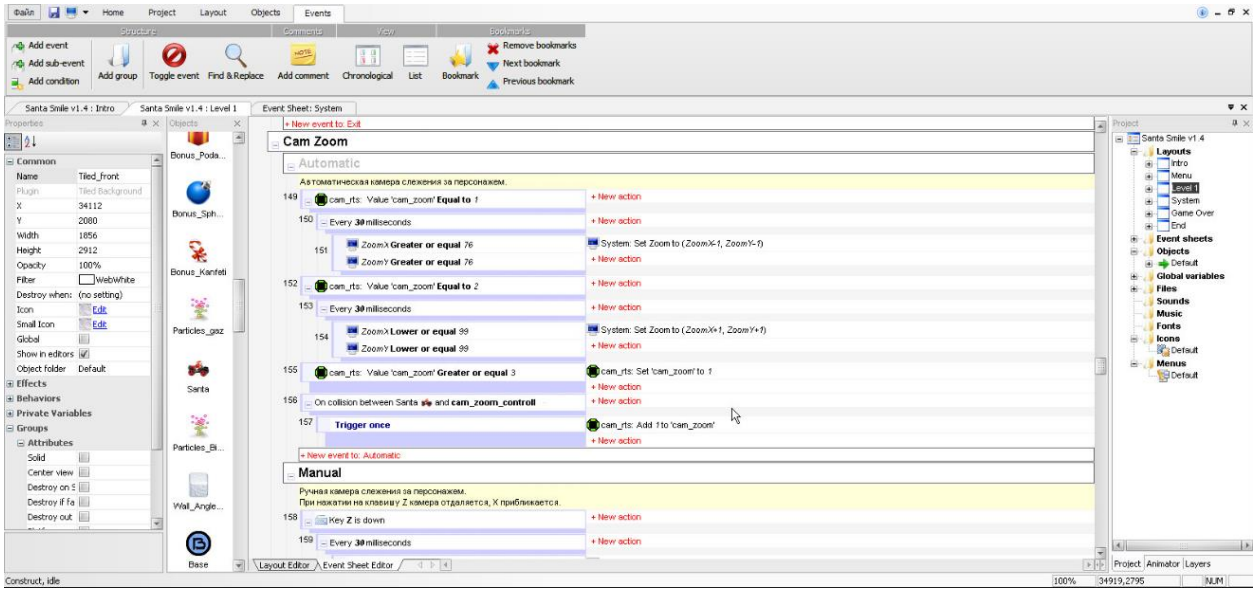

Рис. 1.3

Construct включає в себе більше 60 об'єктів(1.4) і 20 вбудованих(1.5) поводження. За допомогою об'єктів ви створюєте спрайт, керуєте грою, програвайте музику, звуки, створюєте ефекти частинок, і багато багато іншого.

Архітектура об'єктів Construct дає можливість будь-якому охочому C ++ розробнику за допомогою БВК власноруч написати додаткові плагіни і використовувати їх як об'єкти в Construct , що дозволяє безмежно розширювати можливості програми.

Поведінки миттєво розширюють можливості об'єктів. Наприклад, додавши спрайту поведінку Platform(Платформер) ви відразу ж можете управляти їм, бігаючи і стрибаючи по рівню. Ви легко можете міняти швидкість бігу, силу стрибка і т.д., що дозволяє повністю налаштувати поведінку вашого персонажа або будь-якого іншого об'єкта в грі. поведінки економлять час, тому що вони позбавляють вас від необхідності повторно створювати події для однакових завдань. Наприклад, поведінка Fade (Загасання) дає можливість будь-яких об'єктів автоматично зникати або з'являтися на рівні вашої гри. Також поведінки дозволяють управляти ворогами, створювати реалістичну симуляцію фізики і багато іншого.

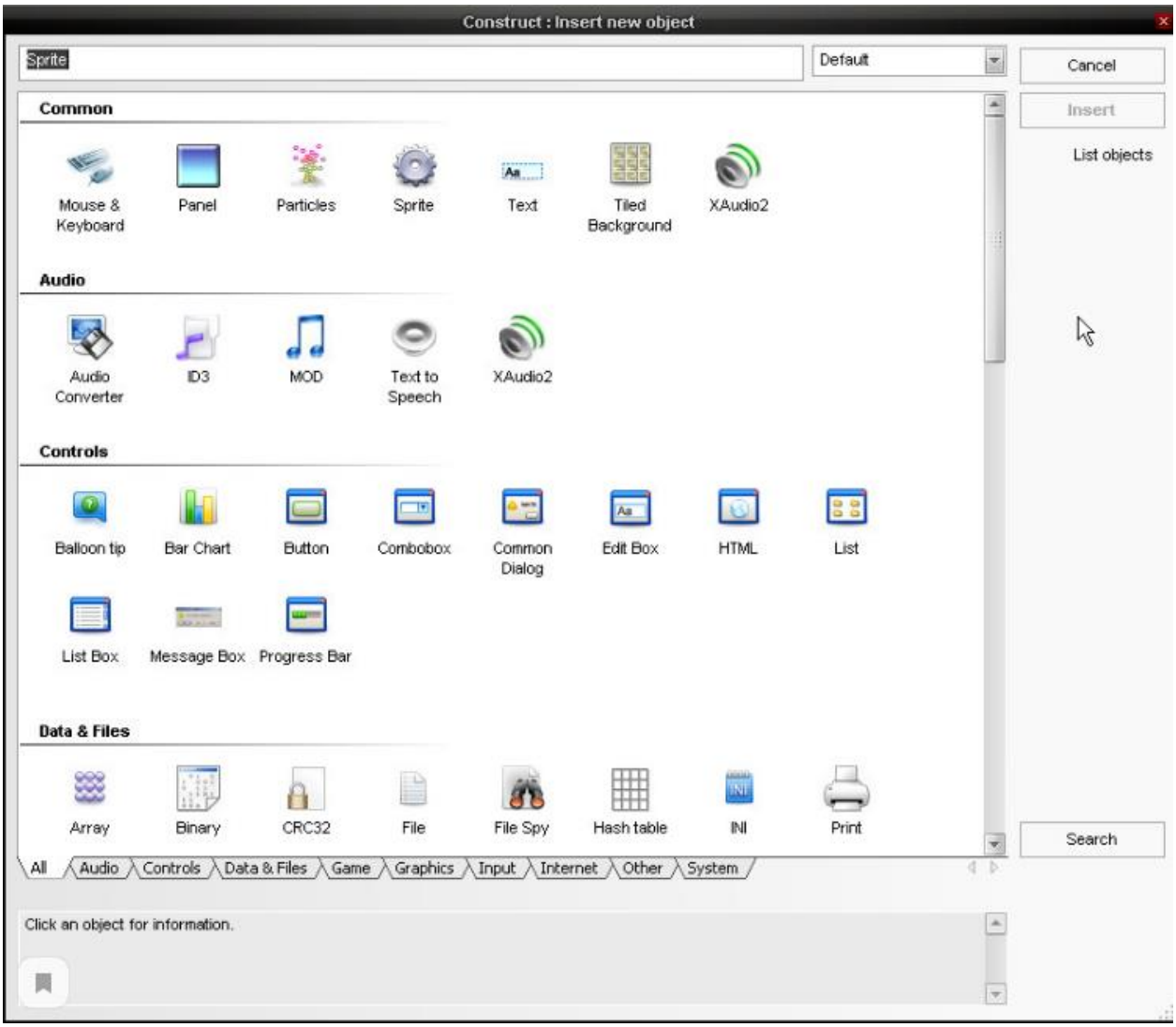

Рис. 1.4

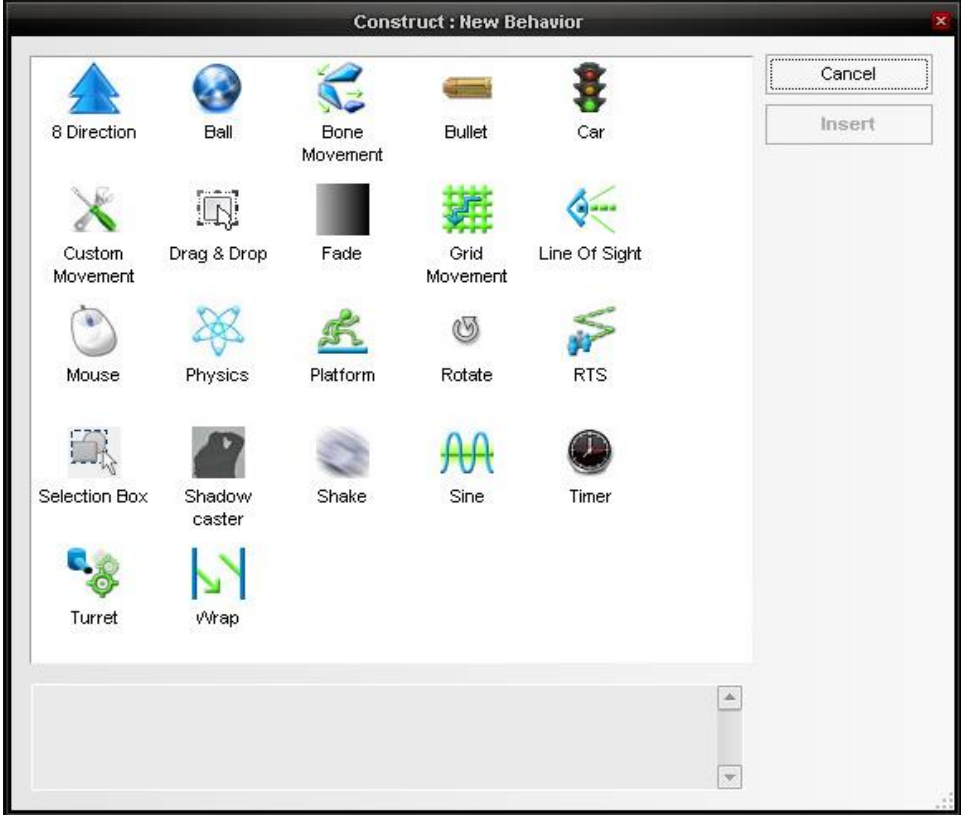

Рис. 1.5

Ефекти і візуалізація

Коли ви запускаєте створену вами гру, візуалізація відбувається за допомогою DirectX9 апаратним прискоренням, що дає оптимальну проізвшітельность за рахунок використання відеокарти, і дозволяє використовувати піксельні шейдери.

У Construct є більше 50 ефектів, за допомогою яких ви можете створювати спотворення, змішування, розмиття, хвильові і лінзові ефекти, маски і багато іншого. Ви можете застосовувати ефекти до певних об'єктам або навіть до цілого рівню. Кожен ефект має гнучкі налаштування, що дозволяє вам швидко і легко створювати красиві і вражаючі спецефекти.

Додаткові функції

Construct має безліч додаткових функцій, які можуть додати ціле новий напрямок в розробці вашої гри. Ось деякі з них:

Поведінка фізики, яке додає реалістичну симуляцію фізики об'єктів.

Система кісткової анімації для створення виключно плавної анімації.

Отладчик для перегляду внутрішнього стану програми під час гри.

3D функції, такі як об'єкти 3D Box і 3D Object для створення ігор з елементами 30 графіки.

Вбудовані сценарії Phyton .

ВИСНОВКИ

Контроль, або перевірка результатів навчання, є обов'язковим компонентом процесу навчання.

Він має місце на всіх стадіях процесу навчання, але особливого значення набуває після вивчення якого-небудь розділу програми або завершення ступені навчання.

Суть перевірки результатів навчання полягає у виявленні рівня засвоєння знань учнями, який повинен відповідати освітньому стандарту за даною програмою, предмету. Однак дидактичні поняття перевірки знань або контролю результатів навчання мають значно більший обсяг в сучасній педагогіці.

Контроль, перевірка результатів навчання трактується дидактикою як педагогічна діагностика, з якою пов'язана проблема вимірювань в педагогіці.

Контроль буде здійснюватися на належному рівні тільки в тому випадку, якщо будуть виконані такі вимоги, як регулярність, всестороннесть, диференційованість, об'єктивність і, звичайно ж, дотримання виховного впливу контролю. Методика викладання в своєму розпорядженні значний теоретичним багажем і практичним досвідом в організації контролю.

Контроль частіше розглядається під кутом зору спрямованості уваги учнів на чисто контролюючу або навчальну його функцію, на смислову сторону.

Проблема контролю постійно привертає до себе увагу вчителів і методистів, так як в ньому криються різноманітні і далеко не вичерпані можливості навчання, що виховує.

Тест - це короткочасне, технічно просто обставлене випробування, проведене в рівних для всіх випробовуваних умовах і має вигляд такого завдання, рішення якого піддається кількісному обліку і служить показником ступеня розвитку до даного моменту відомої функції у даного випробуваного.

Проаналізувавши основні функції та призначення контролю в навчанні, визначивши сутність і особливості розробки тестового контролю, створивши програму конролю. Завдання, поставлені в роботі виконав, тим самим досяг мети дослідження.

СПИСОК ЛІТЕРАТУРИ

1. Горнаков, С.Г. Инструментальные средства программирования и отладки шейдеров в DirectX и OpenGL / С.Г Горнаков. - СПб. : БХВ-Петербург, 2005 (ГУП Тип. Наука). - 248 с. :
2. Маккинли,Уэс. Python и анализ данных [Текст] / Уэс Маккинли ; Пер. с англ. Слинкин А. А. - Москва : ДМК Пресс, 2015. - 482 с.
3. Аванесов, В.С. Форма тестовых заданий : учеб. пособие для учителей шк., лицеев, преподавателей вузов и колледжей / В.С. Аванесов. - [2-е изд., перераб. и расш.]. - М. : Центр тестирования, 2005. - 153, [2] с.; 20 см.; ISBN 5-94635-189-3 : 3000
4. Королев М.Ф., Пашков В.А. Новые методы и средства обучения / Всесоюз. о-во "Знание", Политехн. музей, НИИ пробл. высш. шк. - М. : Знание, 1981. - 88,[1] с.
5. Огородников,Е.В. Резервы системы компьютерных средств обучения в школе : (Естеств.-мат. предметы) / Огородников Евгений Васильевич; Рос. акад. образования, Ин-т средств обучения. - М. : РАО, 1993. - 84 с.
6. Суховиенко,Е.А. Информационные технологии педагогической диагностики: теория и практика : монография / Е. А. Суховиенко ; Гос. образоват. учреждение высш. проф. образования"Челяб. гос. пед. ун-т". - Челябинск : Юж.-Урал. кн. изд-во, 2005. - 238 с. : ил., табл.; 20 см.; ISBN 5-7688-0926-0 : 500
7. В. И. Божич, Горбатюк Н.В.. Интеллектуальная система компьютерного обучения // Перспективные информационные технологии и интеллектуальные системы, № 1 (5), 2001
8. Баев, С.Я. Дидактические основы системы методов теоретического и производственного обучения в профессиональных училищах/С.Я. Баев. – СПб: Нева, 1997. - 400 с.

9. Батышев, С.Я. Профессиональная педагогика: Учебник для студентов, обучающихся по педагогическим специальностям и направлениям. - 2-е изд., перераб. и доп. / С.Я.Батышев. - М.: Ассоциация «Профессиональное образование», 1999. - 904 с.
10. Безрукова, В.С. Педагогика. /В.С. Безрукова.– Екатеринбург: Урал-Е, 1993. – 320 с.
11. Беляева, А.П. Методология и теория профессиональной педагогики/А.П. Беляева. – СПб.: Речь, 1999. – 155 с.
12. Бородина, Н.В., Горонович, М.В., Фейгина М.И. Подготовка педагогов профессионального обучения к перспективно-тематическому планированию: модульный подход/Н.В. Бородина, М.В. Горонович, М.И. Фейгина: -. Екатеринбург: Изд-во Рос.гос.проф.-пед. ун-та, 2002. - 260с.
13. Вишнякова, С.М. Профессиональное образование: Словарь. Ключевые понятия, термины, актуальная лексика/С.М. Вишнякова. - М.: Речь, 1999. – 197 с.
14. Гитман, Е.К. Проектирование содержания специальных дисциплин/Е.К. Гитман //Специалист. – 1997 - № 12 – С. 29.
15. Данилец, Н. А. Ускоренная профессиональная подготовка/ Н.А. Данилец, Т.Е. Колкова, А.Е. Хрупало ; Акад. проф. образования. Журн "Проф. образование". - М. : Изд. центр АПО, 2002. - 42 с.
16. Жуков, Г.Н Основы общей и профессиональной педагогики : Учебное пособие / Под общ ред проф Г.П. Скамницкой. - М.: Гардарики, 2005. - 382 с.
17. Инновации в системе начального профессионального образования: актуал. пробл. проф. подготовки: IX Обл. науч.-практ. конф. (10-11 дек. 2002г.,: Тез. докл. и сообщ. - Челябинск :, 2003 - 151 с.
18. Концепция модернизации российского образования на период до 2010 г//Профессиональное образование. - №4. – 2006. -19 с.

19. Лисовец, А. В. Методы и алгоритмы мониторинга знаний студентов в учебном процессе профессионального образования: 05.13.10: Автореф. дис. на соиск. учен. степ. канд. техн. наук / А.В. Лисовец. - Барнаул : [б. и.], 2002. - 18 с.
20. Макиенко, Н.И. Педагогический процесс в училищах профессионально-технического образования/Н.И. Макиенко. – Минск, Высш. школа, 1977. – 256 с.
21. Никитина, Н.Н., Железнякова, О.М., Петухов, М.А. Основы профессионально-педагогической деятельности/Н.Н. Никитина и др. – М.: Мастерство, 2002 – 288 с.
22. Оконь. В. Введение в общую дидактику/В. Оконь. – М.: Высш. шк., 1990. -340 с.