ЗМІСТ

Скорочення та умовні познаки 3
Вступ 4
1 Дослідження та аналіз архітектури та класів інтелектуальних систем .. 7
1.1 Дослідженні питань створення інтерактивної системи навчання для дітей 7
1.2 Інтелектуальні тьюторські системи 9
1.3 Навчальні середовища з інтелектуальною підтримкою 12
1.4 Інтерактивні репетитори 14
2 Огляд та аналіз типів навчальних сценаріїв 18
2.1 Метод програмованого навчання Скінера 18
2.2 Метод «внутрішнього програмування» Краудера 19
2.3 Сценарно-орієнтовані навчальні системи 21
2.4 Персоналізовані системи навчання 22
2.5 Постановка завдання 24
3 Проектування структури і сценаріїв навчальних епізодів інтерактивної системи 26
3.1 Проектування шаблонів і інформаційного наповнення 26
3.2 Планування типових анімацій в розділах системи 28
3.3 Проектування сценаріїв навчальних фрагментів 31
3.3.1 Навчальний фрагмент «Вгадати клітинку» 32
3.3.2 Навчальний фрагмент «Падаючі фігури» 33
3.3.3 Навчальний фрагмент «Збити всі цукерки» 34
3.3.4. Навчальний фрагмент «Гра на уважність» 36
3.3.5 Навчальний фрагмент «Розташувати фігури» 37
3.3.6 Навчальний фрагмент «Знайти фігури» 39
4 Розробка інтерактивної системи навчання 41
4.1 Вибір програмних засобів реалізації системи 41
4.2 Реалізація сторінок інтерактивної системи навчання 54

4.3 Реалізація анімаційних фрагментів 57
4.4 Розробка ігрових навчальних фрагментів 60
4.4.1 Розробка навчального фрагмента «Вгадати клітку» 60
4.4.2 Розробка навчального фрагменту «Падаючі фігури» 61
4.4.3 Розробка навчального фрагменту «Збити цукерки» 62
4.4.5 Розробка навчального фрагмента «Розставити фігури» 65
4.4.6 Розробка навчального фрагмента «Знайти фігури» 67
Висновки 69
Перелік джерел посилання 70

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

Adobe AIR – це незалежне від платформи операційне середовище для веб-додатків;

App Store – розділ онлайн супермаркету iTunes Store, що продає власникам мобільних телефонів iPhone, планшетних ПК iPad різні додатки;

CSS – Cascading Style Sheets – каскадні таблиці стилів;

DOM – Document Object Model (програмний інтерфейс для доступу до документів);

GameKit – відкритий ігровий движок для платформ Windows, Mac OSX, Linux, Android і iPhone;

HTML – HyperText Markup Language – мова гіпертекстової розмітки

iOS – власницька мобільна операційна система від Apple;

Objective-C – високорівнева об'єктно-орієнтована мова програмування загального призначення;

OpenAL – Open Audio Library – кросплатформовий прикладний програмний інтерфейс (API) для роботи з аудіоданими;

OpenGLES – Open Graphics Library – це відкрита графічна бібліотека;

URL – Uniform Resource Locator.

ВСТУП

Для дітей дошкільного віку найчастіше основною проблемою є складність процесу знайомства зі світом, освоєння необхідної інформації і відсутність постійного інтересу до навчання. Багато батьків не знають, як допомогти дітям краще розвивати свій кругозір, як зробити навчання приємним і цікавим процесом. Внаслідок чого виникають труднощі – відсутність у дитини навичок, знань і умінь, що відповідають вимогам сучасного часу. Однак на даний момент у багатьох сучасних батьків є смартфони та планшети, до яких дитина проявляє інтерес. Виникає питання, чи можуть батьки використовувати їх як засіб для навчання і розвитку своїх дітей за допомогою спеціального додатка здатного організувати навчальний процес в ігровій формі, зробити його простим, доступним і цікавим. Ігрова атмосфера є найбільш підходящим елементом успішного навчального процесу для дітей. Це відповідає передбачуваної концепції програми: набір міні-ігор розвиваючої та освітньої тематики. Додатковим позитивним елементом може стати наявність персонажів-звірят і барвисте оформлення [1][1)].

Аналіз літератури і історія розвитку комп'ютерного навчання виділяє два види навчальних систем – традиційні та інтелектуальні. Основні особливості інтелектуальних навчальних систем полягають в управлінні навчальною діяльністю, з урахуванням всіх її особливостей, на всіх етапах вирішення навчально-пізнавального завдання, починаючи від постановки і пошуку принципу рішення і закінчуючи оцінкою оптимальності рішення. Основними особливостями інтелектуальних навчальних систем також є забезпечення діалогової взаємодії мовою, близькою до природного. В ході діалогу може обговорюватися не тільки правильність дій тих чи інших учнів, а й будується пошук рішення, планування дій, прийоми контролю і т.д. У таких навчальних системах індивідуальне навчання здійснюється на основі динамічної моделі

[1)] [1] Т.О. Піроженко, С.О. Ладивір, К.В. Карасьова. Дитина у сучасному соціопросторі: навчальний посібник. Кіровоград: Імекс-ЛТД, 2014. 272 с.

учня. Системи цього виду дозволяють забезпечити розподіл керуючих функцій між комп'ютером і учнем, передаючи останньому нові навчальні функції, забезпечуючи тим самим оптимальний перехід від навчання до самонавчання. У міру накопичення даних інтелектуальна система може удосконалювати свою стратегію навчання. Діти – це зовсім інша аудиторія, тому необхідний кардинально інший підхід, як щодо графічного дизайну, так і розробки. Діти надзвичайно чутливі і сприйнятливі до всього, що зустрічають в повсякденному житті, і потрібно розуміти, наскільки тонка грань між реальністю і цифровими об'єктами в дитячому світі. З цієї причини пропонований аудиторії контент повинен бути відповідної якості. І якщо у випадку з дорослою аудиторією деякими принципами юзабіліті можна знехтувати, то тут розробник не може дозволити собі подібну розкіш, тому що діти нетерплячі і якщо додаток не доведеться їм до душі, вони більше ніколи його не відкриють [2][1)]. Задовольнити безперервно мінливі бажання і потреби цієї цільової групи досить непросто, але сьогодні планшети і смартфони міцно займають верхні рядки серед бажань дітей будь-якого віку. Тому завдання створення інтелектуального ігрового середовища, зокрема, інтерактивної системи навчання для дітей я є актуальним. На відміну від традиційних комп'ютерних систем, які функціонують на основі закладеного алгоритму, інтелектуальні навчальні системи відповідно до закладених в неї алгоритмами організовують управління навчальною і самостійною діяльністю на евристичному рівні.

Метою магістерської роботи є розробка інтерактивної системи навчання грі в шахи для дітей з використанням інтелектуальних навчальних ігрових сценаріїв для мобільних пристроїв iPad на платформі iOS, що забезпечить отримання навичок гри в шахи в ігровій формі.

[1)] [2] Новик І.М. Проектування навчальних компютерних ігор в освітньому процесі дошкільного навчального закладу – PSYH.KIEV.UA – Вісник психології і соціальної педагогіки. URL: https://www.psyh.kiev.ua/Новик_І.М._Проектування_навчальних _компютерних_ігор _в_освітньому_процесі_дошкільного_навчального_закладу (дата звернення: 15.09.2019).

Для досягнення поставленої мети, необхідно вирішити в рамках виконання магістерської роботи наступні завдання:

- провести дослідження архітектурних рішень та класів інтелектуальних систем навчання;
- здійснити дослідження, аналіз та вибір типів сценаріїв навчання;
- провести проектування структури, сценаріїв та епізодів інтерактивної системи навчання для дітей;
- здійснити вибір середовища, фреймворка та засобів реалізації інтерактивної системи навчання;
- розробити типові анімації для розділів гри в шахи;
- здійснити розробку фрагментів ігрового навчального середовища для дітей.
- здійснити програмну реалізацію інтерактивної системи навчання для дітей на платформі iOS.

Застосування інтерактивної системи навчання грі в шахи для дітей надасть максимально бажаний ефект – засобами мобільного пристрою на платформі iOS, в ігровій формі дитина самостійно отримує навички гри в шахи. Регулярні заняття з освоєння гри в шахи навчать маленького шахіста розумно мислити, розсудливо і творчо підходити до вирішення найскладніших завдань, швидко і правильно вживати заходів.

Розробка інтерактивної системи навчання грі в шахи засновано на матеріалах однойменної книги «Шахматы для малышей» [3][1)].

Магістерська робота містить 76 сторінок, 29 рисунків, 15 посилань.

[1)] [3] Костина Ю.К. Шахматы для малышей. Х.: Фактор, 2012. 96 с.

1 ДОСЛІДЖЕННЯ ТА АНАЛІЗ АРХІТЕКТУРИ ТА КЛАСІВ ІНТЕЛЕКТУАЛЬНИХ СИСТЕМ

1.1 Дослідженні питань створення інтерактивної системи навчання для дітей

Для здійснення проектування та розробки інтерактивної системи навчання для дітей необхідно перш за все здійснити дослідження та аналіз існуючих архітектур та класів інтелектуальних систем та здійснити вибір потрібної і більш відповідної поставленій меті архітектури та обрати клас інтелектуальної системи.

Еволюція систем навчання широко відображає розвиток різних класів інтелектуальних систем навчання – систем індивідуального навчання, які називаються також – інтелектуальні тьюторські системи.

Однак, навчання в процесі спілкування з тьютором не є єдиним процесом, що забезпечує придбання нових знань. У теорії навчання відомий ряд інших способів придбання знань. Кожному з цих способів відповідає свій клас інтелектуальних систем навчання. Для кращого розуміння даного питання, все інтелектуальні системи навчання можуть бути розділені на три класи [4][1)]:

– інтелектуальні тьюторські системи;

– навчальні середовища з інтелектуальною підтримкою;

– інтелектуальні репетитори.

Не дивлячись на різну філософію процесу навчання і передачі знань перерахованих систем, у них є щось спільне – їх архітектура базується на однакових компонентах: модель предметної області (експертні знання про предметну область), модель якого навчають, стратегія навчання, або експертні знання про метод навчання та інтерфейс навчальна система – це той, якого

[1)] [4] И.А. Чмырь, М.Ф. Ус, А.В. Пискун. Интелектуальные системы обучения. Конспект лекций. Одесса: Издательский центр ОГАХ, 2000. 110 с.

навчають. На рис. 1.1 приведена уніфікована архітектура інтелектуальних систем навчання.

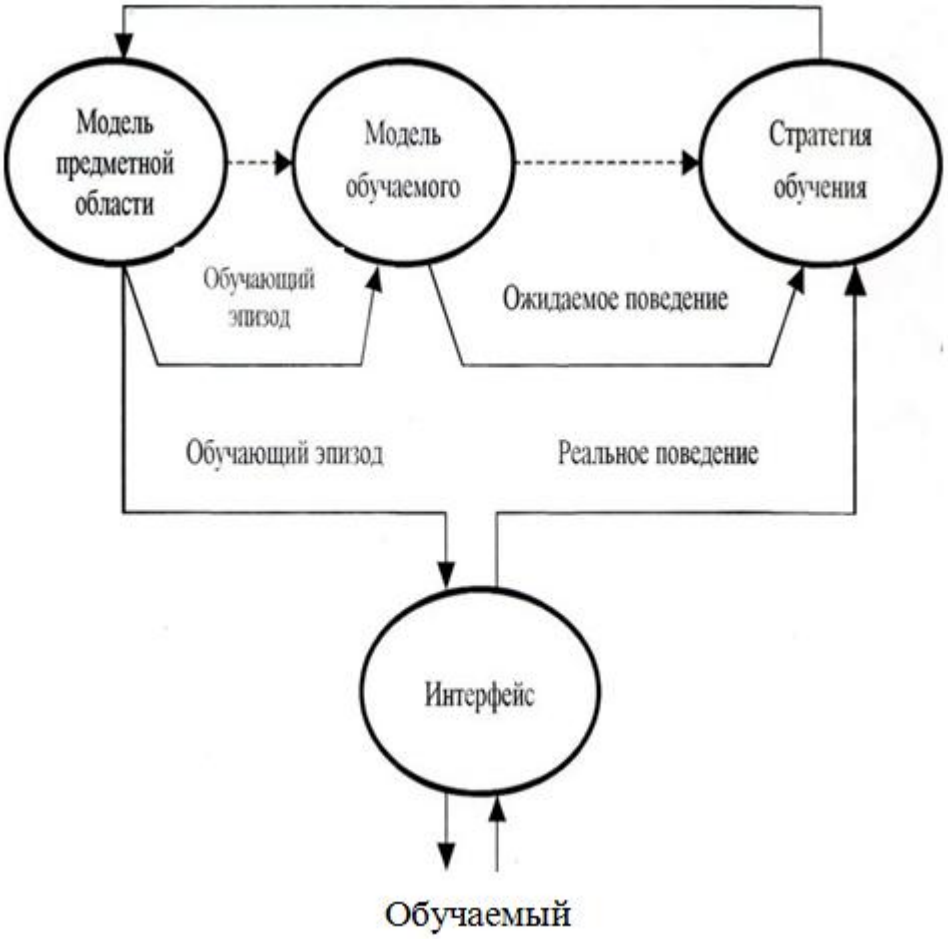

Рисунок 1.1 – Уніфікована архітектура інтелектуальних систем навчання

Розглянемо більш детально основні компоненти інтелектуальних систем навчання.

Модель предметної області (domain knowledge, domain expertise) виконує подвійну функцію. З одного боку, вона є джерелом предметних знань, які представляються тому, якого навчають, а з іншого – це бажаний стандарт знань учня (або мета навчання), який можна використовувати для оцінки поточних знань. Модель учня (student model) призначена, головним чином, для прогнозування поведінки учня. Ідеальна модель повинна робити точні прогнози поведінки будь-якого учня в будь-якому контексті предметної області.

Наприклад, у вигляді безлічі можливих реакцій на запропонований навчальний стимул для випадку інтелектуальної тьюторської системи.

Стратегія навчання (pedagogical expertise), для випадку інтелектуальної тьюторської системи, визначає який навчальний стимул і в який момент часу повинен бути переданий тому, якого навчають. У найбільш загальному випадку стратегія навчання не кодують жорстко, а виводиться з поточної ситуації та методу, що зберігається у вигляді набору принципів або правил. Дидактичні рішення приймаються на основі передбачення реакції учня, що генерується моделлю учня і його реальної реакцією. Стратегії навчання варіюються по відношенню до ступеня «свободи» надається тому, якого навчають і можуть бути впорядковано розташовані між двома полюсами: від повного контролю усіх дій учня і виконання «лоцманських» функцій при проведенні якого навчають через безліч навчальних стимулів, до повної свободи навігаційної активності учня і видачі рекомендацій тільки за його запитом [4][1)].

Інтерфейс навчальної системи учня являє собою або варіант обмеженої природної мови, або різні графічні засоби спілкування.

1.2 Інтелектуальні тьюторські системи

Інтелектуальні тьюторські системи являють собою найбільш великий клас електронних систем навчання. Головна ідея цих систем – імітація навчального поведінки досвідченого тьютора – людини [5][2)].

Ідеальну інтелектуальну тьюторські систему можна описати таким чином. Система, в процесі взаємодії з учнем, безперервно «підлаштовується» до його поточного рівня знань і ступеня розуміння, вибирає навчальний матеріал, для якого має місце невідповідність в рівні знань і / або ступеня розуміння, автоматично доповнює навчальний матеріал тематично пов'язаної інфор-

---

[1)] [4] И.А. Чмырь, М.Ф. Ус, А.В. Пискун. Интелектуальные системы обучения. Конспект лекций. Одесса: Издательский центр ОГАХ, 2000. 110 с.

[2)] [5] В.Д. Шарко. Інтерактивні методи навчання: Досвід впровадження. Херсон: Олді-Плюс, 2000. 210 с.

мацією (наприклад, довідкового характеру), приділяє більше уваги «важким місцям» і пропускає вже відомий матеріал, тестує учня і визначає його когнітивні характеристики, проводить процес навчання методом, відповідним поточним когнітивним здібностям і стилю учня.

Розглянемо характерні особливості і приклади двох найбільш відомих інтелектуальних тьюторських систем. Однією з перших, на поточний момент вже класичних, інтелектуальних тьюторських систем є SCHOLAR, запропонована Карбонеллом в 1970 році і призначена для навчання студентів університетів географії Південної Америки. Інтерфейс навчальної системи – це той, якого навчають в SCHOLAR забезпечується діалоговим процесом, в якому ролі партнерів не зафіксовані. Це означає, що під час навчання питання може задавати не тільки система, а й той, якого навчають. Для спілкування з учнем SCHOLAR використовує англійську мову. Генеруються питання, і фрази не є довільними, а створюються за допомогою шаблонів, що доповнюються інформацією з моделі предметної області. Процедура генерації не припускав «розуміння» природної мови. SCHOLAR це система, де вперше для моделювання предметної області використовувалися семантичні мережі. Автори SCHOLAR покладали великі надії на семантичні мережі і ввели поняття «модель ідеальних знань учня» або «модель ідеального студента» (perfect student), як синонім поняття предметної моделі. Під моделлю ідеального студента в SCHOLAR розумілася модель предметної області у вигляді повної семантичної мережі [4][1)].

Для відповідей на питання учня, якого навчають SCHOLAR використовує механізм логічних умовиводів. Техніка логічних умовиводів, в більшості випадків, зводиться до навігації по вузлах мережевої моделі предметної області. Наприклад, система може легко прийти до висновку, що Сантьяго знаходиться в Південній Америці оскільки вузол «Сантьяго» пов'язаний з вузлом «Чилі» ставленням приватне-ціле, який в свою чергу пов'язаний тим же

[1)] [4] И.А. Чмырь, М.Ф. Ус, А.В. Пискун. Интелектуальные системы обучения. Конспект лекций. Одесса: Издательский центр ОГАХ, 2000. 110 с.

відношенням з вузлом Південна Америка. Приклад фрагмента моделі предметної області SCHOLAR наведено нижче на рисунку 1.2.

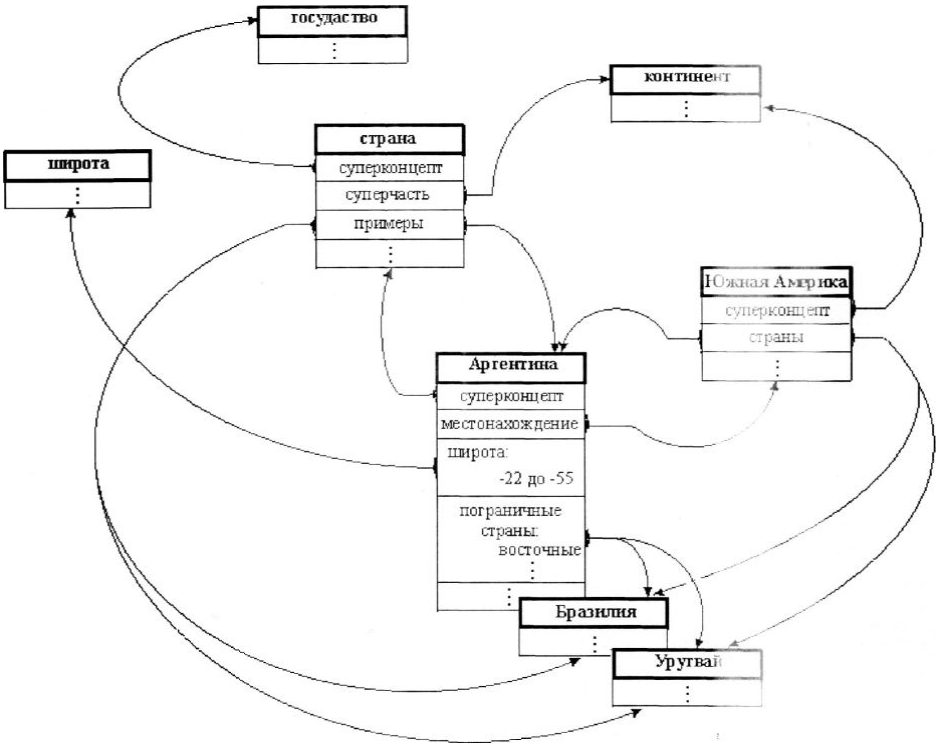

Рисунок 1.2 – Фрагмент мережевої моделі предметної області SCHOLAR

Розглянемо ще одну досить відому інтелектуальну тьюторську систему WHY. Дана система розроблена Стівеном і Коллінсом, в кінці 70-х і початку 80-х, характерна тим, що в ній вперше була зроблена спроба комп'ютерної реалізації «некомп'ютерної» стратегії навчання, відомої як метод Сократа (Socratic method). WHY значно спрощують процес навчання метеорології студентами вищих навчальних закладів. Метод Сократа є суто діалоговим і передбачає такий спосіб ведення діалогу, при якому той, якого навчають, «ведений» питаннями тьютора, повинен за допомогою дедуктивних умовиводів сам виводити нові, невідомі йому раніше, відносини між відомими фактами предметної області. Метод Сократа, таким чином, підкоряється деяким загальним правилам, які визначені стратегію навчання. Коллінс і його колеги запропонували спочатку кілька десятків, а потім понад півсотні правил сфо-

рмулювати у вигляді умовних пропозицій, і визначають поведінки аматора, що використовує метод Сократа, стосовно предметної області WHY. Кожне правило пов'язує один з очікуваних поточних відповідей навчаємого о з наступним навчальним стимулом тьютора [4][1)].

З наведеного визначення процесу навчання слід стратегія навчання, заснована на постійному зіставленні модельних передбачень з поведінкою реальної фізичної системи, що є основною технікою методу Сократа. Невідповідності, виявлені в процесі діагностики, мають служити причиною коригування ментальної моделі учня, а оскільки між ментальними моделями існує ієрархічна впорядкованість, навчання може розглядатися також як процес діагнозування корекції ментальних моделей учня на все більш глибоких рівнях.

1.3 Навчальні середовища з інтелектуальною підтримкою

При проектуванні навчальних середовищ з інтелектуальної підтримкою не ставиться за мету розробити «штучний тьютор», який веде учня через матеріал предметної області та враховує його когнітивну індивідуальність. Навчальне середовище являє собою систему, призначену для навчання шляхом експериментування. Як правило, метод навчання, який використовується в інтелектуальних навчальних середовищах відноситься до філософії спрощеного конструктивізму і полягає в тому, що навчання здійснюється в процесі активної взаємодії з деякою, спеціальним чином підготовленої, навчальної середовищем. Той, якого навчають експерт працюючи з середовищем перевіряє свої ідеї та гіпотези і отримує у відповідь критичні зауваження і поради. Методи навчання, що базуються на ідеях конструктивізму, призначені, в першу чергу, для отримання процедурних знань, або навичок.

Так, наприклад, при вивченні іноземних мов методом «занурення», учню, якого навчають, може бути надано штучне середовище, яким він може

[1)] [4] И.А. Чмырь, М.Ф. Ус, А.В. Пискун. Интелектуальные системы обучения. Конспект лекций. Одесса: Издательский центр ОГАХ, 2000. 110 с.

маніпулювати за допомогою маніпулятора типу «миша». Наприклад, перебуваючи в середовищі ванної кімнати навчають, може відкрити кран і наповнити водою раковину, взяти шматочок мила і намочити його і т.д. Будь-які дії учня супроводжуються коментарями системи на мові, що вивчається як у вигляді тексту, так і шляхом аудіювання. Система може також, використовуючи мову навчання, пропонувати учню виконання деяких дій. На рисунку 1.3 наведено фрагмент навчального середовища, що використовується для навчання іспанської мови в навчальній системі FLUENT (Foreign Language Understanding T gendered by Natural Techniques). Особливістю інтерфейсу навчальної системи в FLUENT – широке застосування графічних сцен, які супроводжуються вербальними командами або коментарями [4][1)].

Рисунок 1.3 – Приклад навчальної сцени у навчальному середовищі FLUENT

Класичним прикладом навчального середовища з інтелектуальною підтримкою є система SOPHIE, розроблена Брауном і його колегами в середині 70-х і початку 80-х років. SOPHIE призначена для навчання навичкам пошуку дефектів в електронних схемах. Розроблено кілька версій SOPHIE. Від-

[1)] [4] И.А. Чмырь, М.Ф. Ус, А.В. Пискун. Интелектуальные системы обучения. Конспект лекций. Одесса: Издательский центр ОГАХ, 2000. 110 с.

мінною особливістю всіх версій є досконалий природно-мовний інтерфейс, який спілкується на теорії семантичного розбору пропозицій.

Єдине, що робить система SOPHIE, в сенсі стратегії навчання – це фіксує дефект і пропонує тому, кого навчають його виявити. Наслідком цього є розподіл ролей партнерів діалогу. Розглянемо приклад.

Навчаємий: Now what is the outpt? SOPHIE: (outpt => output) the output voltage is 1.2 volts. If you are not sure whether a value is correct, can ask "is that value correct?". Після того як помилка в слові «outpt» розпізнає і наявна невизначеність дозволена як «output voltage» SOPHIE інформує учня про одну зі своїх можливостей.

Навчаємий: Is that right? SOPHIE: in a working circuit the output voltage is 30.1 volts. The value is incorrect. Висновок про анафору: слово «that» відноситься до величини попереднього виміру.

Навчаємий:: What is the voltage between the current limiting transistor and the constant current source? SOPHIE: the voltage at №4 is 2.48 volts. Слово «between» інтерпретується як посилання на певну точку в якій вимірюється напруга по відношенню до землі.

Навчаємий: What is it between there and the base of Q6? SOPHIE: the voltage between №4 and №5 is 1.28 volts. Слово «it» відноситься до напруги, а слово «there» – до №4. Слово «between» в цьому випадку інтерпретується не так як в попередньому питанні

SOPHIE, слідуючи філософії спрощеного конструктивізму, відрізняється тим, що не проявляє ніякої педагогічної ініціативи і, тому, не містить стратегію навчання.

1.4 Інтерактивні репетитори

Головна мета інтелектуального репетитора полягає в підтримці навчальної активності учня, при виконанні завдання, шляхом трансформації скоєних ним помилок в процедурний досвід. Репетитор не втручається в процес

вирішення завдання до тих пір, поки не виявляє, помилкові дії. Таким чином, репетитор здійснює безперервне і непомітне відстеження активності учня і перериває його роботу тільки тоді, коли дії учня не збігаються з діями, очікуваними системою. Мистецтво репетиторства полягає в знаходженні найкращого моменту переривання активності учня і найбільш коректного формулювання підказки.

Програми WIJSOR є репетиторами комп'ютерної гри WUMPUS, відомої тим, що вона часто використовується як предметна область при дослідженні, і вивченні процесу логічних умовиводів в прикладних системах штучного інтелекту. Ця комп'ютерна гра проводить гравця через послідовність печер в лабіринті. Гравця чекають різні небезпеки: чудовисько WUMPUS, бездонні ями і кажани, які підхоплюють гравця і кидають його в довільну печеру. Як тільки гравець потрапляє в нову печеру, він отримує список прилеглих печер і набір ознак, що дозволяють зробити висновок про можливу небезпеку: протяг або писк, що попереджають про яму або кажана в одній із суміжних печер; поганий запах, який свідчить про те, що на відстані не більше ніж в два печери знаходиться wumpus. Гравець переміщається шляхом вибору однієї з суміжних печер. Для перемоги в грі необхідно вистрілити однієї з п'яти стріл в лігво wumpus. Гру програно, якщо гравець провалюється в яму, або потрапляє в лігво з wumpus, або витратить всі стріли. Рішення про те, яка з навколишніх печер є безпечною і, отже, може бути обрана такою, вимагає від гравця проведення логічних умовиводів. Перша версія репетитора WUSOR-I включає два модулі: «експерт» і «порадник». Модуль «експерт» містить опис предметної області у вигляді набору правил продукції, що зв'язують ознаки і небезпеки, і дозволяють отримати прогноз про небезпеку або безпеку суміжних печер. Модуль «порадник» не зберігає стратегію навчання. Він втручається в гру щоразу, коли учень робити не оптимальний вибір наступної печери і надає йому набір відповідних пояснень [4][1)].

[1)] [4] И.А. Чмырь, М.Ф. Ус, А.В. Пискун. Интелектуальные системы обучения. Конспект лекций. Одесса: Издательский центр ОГАХ, 2000. 110 с.

Головна відмінність WUSOR-II від WUSOR-I – це наявність моделі учня, що базується на теорії оверлею (overlay theory) Кара і Гольдштейна. Оверлей Кара і Гольдштейна є класичною парадигмою і часто використовується при моделюванні учня. Теорія оверлею передбачає оцінку знань учня шляхом порівняння його поведінки з поведінкою експерта. При цьому приймається, що знання учня є підмножиною знань експерта. Програма WUSOR-III відрізняється від WUSOR-II ще більшим акцентом на подання знань, орієнтоване на конкретного учня. При проектуванні WUSOR-III була запропонована концепція генетичного графа (genetic graph). Генетичний граф є графічним представленням еволюційних, відносин між фрагментами знань і базується на еволюційної теорії інтелекту Піаже. Вузли генетичного графа відповідають елементарним навикам, (представленим індивідуальним правилом), а гілки – їх еволюційним відносин, таким як узагальнення або аналогія. У WUSOR-IIJ модель учня представлена оверлеєм вузлів генетичного графа, а індивідуальна історія якого навчають представлена оверлеєм мереж генетичного графа.

Генетичний граф може розглядатися як уніфікована мережева (сценарна) модель, що включає як стратегію навчання, так і стратегію діагностики і надає кошти персоналізованого підходу до навчання. Серія програм, під загальним найменуванням «когнітивний тьютор», розроблених в університеті Карнегі Мелон (США) є одним з найбільш відомих прикладів комп'ютерних репетиторів. Головним ідеологом цих проектів є Джон Андерсон, який розробив кілька версій уніфікованої теорії пізнання, відомої під назвою ACT (Adaptive Control of Thought). Когнітивні тьютори розроблялися з метою перевірки ключових ідей ACT, головною з яких є переконання в тому, що процедурні знання в пам'яті людини зберігаються у вигляді системи продукці Програма WUSOR-III відрізняється від WUSOR-II ще більшим акцентом на подання знань, орієнтоване на конкретного учня. При проектуванні WUSOR-III була запропонована концепція генетичного графа (genetic graph). Генетичний граф є графічним представленням еволюційних, відносин між фрагментами знань і базується на еволюційної теорії інтелекту Піаже. Вузли генетич-

ного графа відповідають елементарним навикам, (представленим індивідуальним правилом), а гілки – їх еволюційним відносин, таким як узагальнення або аналогія. У WUSOR-IIJ модель учня представлена оверлеєм вузлів генетичного графа, а індивідуальна історія якого навчають представлена оверлеєм мереж генетичного графа [6][1].

Стратегія навчання, яка використовується в цих репетиторів, називається "трасування моделі" (model tracing) і, нагадує класичний принцип регулювання по відхиленню в системах автоматичного регулювання. Репетитор зберігає еталон процедурних знань, які повинен освоїти той, якого навчають, у вигляді послідовності кроків-дій. У тому випадку, якщо фактична послідовність кроків-дій учня збігається з еталонною, репетитор не втручається в дії учня. Якщо дії учня відхиляються від еталонних, то репетитор реагує висновком повідомлення, що містить розшифровку помилки і рекомендації щодо подальших дій. Репетитор містить безліч варіантів помилок учня і тому в змозі давати персоналізовані рекомендації. Як при описі зразка, так і при описі можливих відхилень використовується система продукційних правил, впорядкованих ієрархією цілей та правил, впорядкованих ієрархією цілей.

[1)] [6] Кашев С.С. Технология интерактивного обучения. Мн.: Белорусский верасень, 2005. 196 с.

2 ОГЛЯД ТА АНАЛІЗ ТИПІВ НАВЧАЛЬНИХ СЦЕНАРІЇВ

2.1 Метод програмованого навчання Скінера

Історично першим машинно-орієнтованим методом індивідуального навчання, який поклав початок всієї науки про машинному освіту, є метод, запропонований Скінером в середині 50-х років і названий пізніше метод програмованого навчання. Тип сценарію, запропонованого Скінером, прийнято називати лінійної програмою. Лінійна програма характеризується наступними відмітними рисами.

Матеріал розбивається на прості порції, (навчальні стимули) засвоєння змісту яких не становить труднощів для учня. У разі, коли навчальний стимул представлений вербально, він складається з утвердження і питання, пов'язаного з твердженням. На кожен навчальний стимул система очікує дві відповіді: позитивну і негативну. Незалежно від того, яку отримано відповідь (позитивну або негативну) здійснюється перехід до наступного стимулу. У деяких системах перехід наступного стимулу здійснюється в разі позитивної відповіді, а в разі негативної відповіді – повторюється поточний стимул. Той, якого навчають отримує негайну оцінку своєї відповіді: «вірно», або «невірно». Послідовність стимулів формується за правилом «від простого до складного».

Кількість нових відомостей в кожному черговому стимулі не велика. Це робиться для того, щоб в процесі діалогу кількість невірних відповідей не перевищувало 5% від загальної кількості відповідей.

У цьому випадку забезпечується постійний рівень позитивних емоцій учня. Скінеровскі лінійні програми можуть використовуватися не тільки для навчання, але і для тестування. Для цього, по-перше, необхідно вимірювати характеристики відповідей учня і реєструвати отримані значення, а по-друге, відображати отримані оцінки на шкалі вихідних балів. Вимірюваними харак-

теристиками можуть бути, наприклад, час обмірковування і кількість невірних відповідей [4][1)].

На рисунку 1.4 зображений фрагмент сценарію, відповідний загальному випадку лінійної програми у вигляді графа мережі Петрі, де Si – перший навчальний стимул; RP, RN – позитивні і негативні відповіді, відповідно.

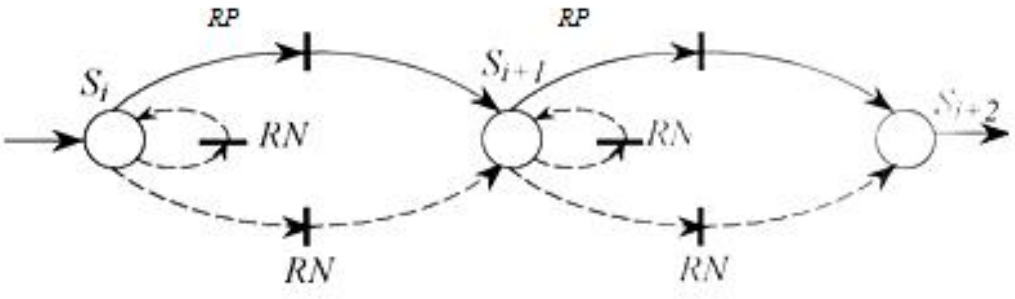

Рисунок 1.4 – Графічне представлення сценарію, відповідно до лінійної програми Скінера

Дидактична цінність лінійної програми визначається не складною і розгалуженою «логікою» сценарію, а тим, наскільки вдало втілений принцип «від простого до складного» в ланцюзі навчальних стимулів (наскільки вдало розподілений матеріал по ланцюгу навчальних стимулів).

2.2 Метод «внутрішнього програмування» Краудера

Наступним, після скінеровської послідовності, типом сценарію, в порядку зростання структурної складності його організації, є метод, запропонований Краудером на початку 60-х років. Цей метод часто називають «внутрішнім програмуванням». В основі методу Краудера лежить можливість отримання більш двох (позитивна і негативна) відповідей на пред'явлений

[1)] [4] И.А. Чмырь, М.Ф. Ус, А.В. Пискун. Интелектуальные системы обучения. Конспект лекций. Одесса: Издательский центр ОГАХ, 2000. 110 с.

навчальний стимул. Серед очікуваних відповідей одна – позитивна (правильна відповідь), а решта – негативні. Отримання будь-якої відповіді означає перехід до наступного стимулу. У разі позитивної відповіді, учень отримує стимул, який містить порцію нових знань, і просувається по головній послідовності стимулів. У разі негативної відповіді, наступний стимул має інформацію, що роз'яснює помилку, і допомагає відповісти правильно. Потім, учню знову пропонується попередній стимул з головної послідовності. Відповідно до методу Краудера негативні відповіді повинні допускатися і, навіть, спеціально передбачатися, оскільки вони дозволяють сформувати індивідуальну послідовність стимулів для конкретного учня. При цьому кожну негативну відповідь повинно бути роз'яснено учню одразу ж після її отримання [7][1)].

На рис. 1.5 зображений фрагмент сценарію, побудованого відповідно до принципів Краудера у вигляді графа мережі Петрі

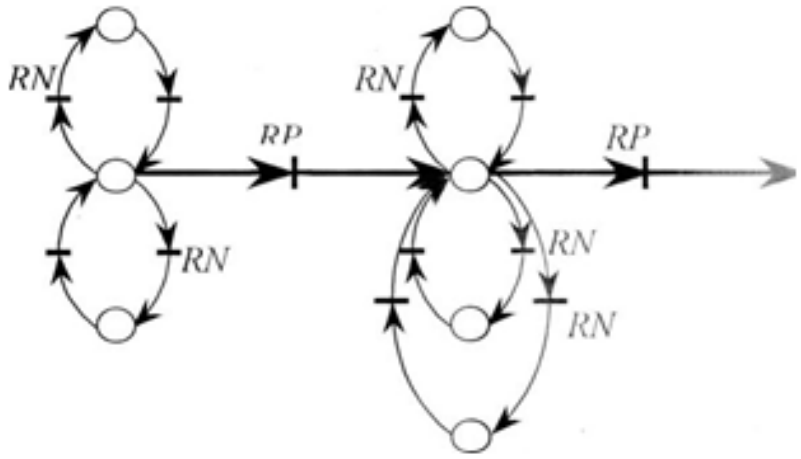

Рисунок 1.5– Графічне представлення сценарію, відповідно до принципів Краудера

У розглянутих у попередньому розділі сценаріях типу «скінеровська послідовність» всі учні отримують одну і ту ж послідовність навчальних

1) [7] Аверин В.А. Психология детей и подростков. СПб: Изд-во В.А. Михайлов, 1998. 379 с.

стимулів незалежно від їх індивідуальних особливостей, тому що метод Скінера спирається на загальні закономірності процесу навчання, тому вони і не залежать від індивідуальних особливостей.

У сценаріях, побудованих за методом Краудера кожен учень, як правило, отримує свою індивідуальну послідовність стимулів. Тому можна говорити, що сценарій, спроектований за методом Краудера може враховувати індивідуальні особливості учня. Однак це справедливо лише для стимулів, що настають за негативними реакціями. Головну послідовність стимулів отримують всі учні.

2.3 Сценарно-орієнтовані навчальні системи

Розвиток технології лазерних носіїв інформації призвело до появи окремого класу сценарно-орієнтованих навчальних систем – електронні книги. Стимули-сторінки електронної книги, як правило, пов'язані в сценарій стандартним чином незалежно від її змісту і тому, сценарій електронної книги часто представлений у вигляді сценарних блоків з однаковою топологією. На рис. 1.6 приведена організауція типового сценарного блоку електроної книги.

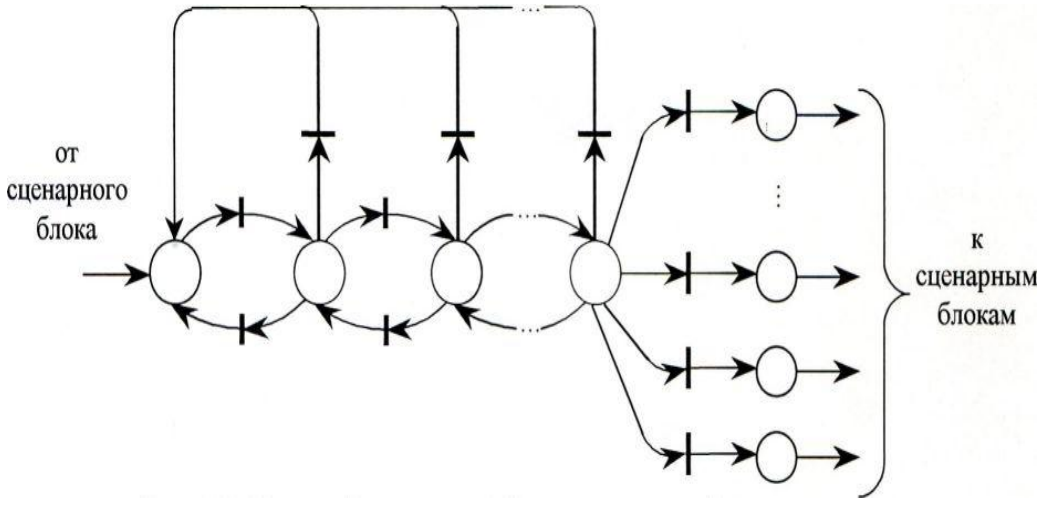

Рисунок 1.6 – Графічне представлення сценарію для електронної книги

Використовуючи термінологію Краудера, відзначимо, що в електронній книзі всі навчальні стимули-сторінки об'єднані в головну послідовність, а всі відповіді позитивні і являють собою команди управління такі як [4][1]:

– наступна сторінка;

– попередня сторінка;

– до початку блоку;

– вибрати пункт меню і т.д.

Таким чином, в разі застосування застосування сценарно-орієнтованої навчальної системи – електронної книги, діалоговий процес вироджується в директивну інтерактивну взаємодію із стандартизованим, в межах конкретної електронної книги, інтерфейсом.

2.4 Персоналізовані системи навчання

Розглянемо, яким чином, при генерації чергового навчального стимулу враховуються попередні відповіді учня, або (використовуючи термін «зворотний зв'язок»), яким чином реалізується зворотний зв'язок від учня до сценарію. Сценарій типу «скінеровська послідовність», очевидно, має найбільш просту організацію. Черговий навчальний стимул генерується без урахування отриманої відповіді. Отже, зворотний зв'язок від учня до сценарію відсутній, а принцип генерації чергового навчального стимулу може бути сформульовано таким чином: «якою б відповідь не була отримана від того, якого навчають, йому необхідно передати черговий стимул з послідовності».

Сценарій типу «електронна книга» має кілька більш складну організацію. Тут є неявна зворотний зв'язок в тому сенсі, що можливість отримання чергової сторінки визначається місцезнаходженням в сценарії, але надається самому, якого навчають [5][2].

---

[1] [4] И.А. Чмырь, М.Ф. Ус, А.В. Пискун. Интелектуальные системы обучения. Конспект лекций. Одесса: Издательский центр ОГАХ, 2000. 110с.

[2] [5] В.Д. Шарко. Інтерактивні методи навчання: Досвід впровадження. Херсон: Олді-Плюс, 2000. 210 с.

Сценарій, розроблений за методом Краудер, явно враховує зворотний зв'язок від учня до сценарію. Казуальна імплікація, закладена в сценарій Краудер, може бути описана умовними пропозиціями двох типів:

– ЯКЩО отримано позитивну відповідь, ТО передати черговий навчальний стимул з головної послідовності;
– ЯКЩО отримано негативну відповідь, ТО передати стимул-роз'яснення І запропонувати повторну спробу.

Використаний в методі Краудера зворотний зв'язок дуже примітивно враховує індивідуальні особливості учня, а його когнітивні здібності повністю ігноруються. Це очевидно, оскільки сценарій, побудований за методом Краудера, запропонує будь-якому учню, одну і ту ж головну послідовність. Чим більшою мірою сценарій враховує індивідуальні особливості учня, тим цінніше, з дидактичної точки зору, є навчальна система. Сценарій, що враховує індивідуальні особливості учня, повинен характеризуватися такими ознаками:

– є зворотній зв'язок від учня до сценарію;
– зворотний зв'язок використовується для визначення індивідуальних характеристик учня;
– сценарій пропонує учню головну послідовність, відповідну його індивідуальним характеристикам.

Система навчання, що володіє перерахованими властивостями, називається персоналізованої системою навчання. Отже, з визначення персоналізованої системи навчання, її сценарій повинен містити не одну, а безліч головних послідовностей навчальних стимулів. Це необхідно для того, щоб кожен учень зміг працювати з індивідуальної головною послідовністю.

Персоналізовані системи навчання, в основному своїй більшості, розраховані на самостійну роботу учнів у власному темпі і орієнтовані на повне засвоєння змісту навчального матеріалу з умовою переходу до нового після засвоєння попереднього. На сьогоднішній день, такого ефекту можна доби-

тися, використовуючи різні індивідуальні мобільні пристрої, такі як мобільні телефони, смартфони, планшети і т.д.

Беручи до уваги той факт, що метою даної магістерської роботи є розробка програми на мобільні пристрої з персоналізованої системою навчання, перейдемо до постановки завдання що до розробки інтерактивної системи навчання для дітей на платформі iOS, а саме визначимо функції, які повинна виконувати система.

2.5 Постановка завдання

Завданням даної магістерської роботи є розробка мобільного додатка інтерактивної системи навчання для дітей на платформі iOS – розробка інтерактивної книги для дітей, по якій вони зможуть навчатися грі в шахи. В якості основи для реалізації була обрана книга «Шахматы для малышей» автора Юлії Костіной. Для поліпшення засвоєння матеріалу необхідно розробити практичні завдання, які допоможуть дітям краще засвоїти знання про шахівницю, знання про те як ходять фігури і загальні правила гри в шахи.

Розглянемо вимоги до інтерфейсу мобільного додатка. При першому запуску користувач повинен побачити основні елементи управління. Такими елементами є: кнопка переходу в меню, перегортання вперед і перегортання назад. Слід зазначити, що перегортання має здійснюватися з імітацією перегортання книги, враховуючи той факт, що користувачами додатку будуть діти. Для кращого засвоєння матеріалу дітьми, після кожного розділу необхідно організувати тематичні завдання. При розробці ігрових фрагментів будемо враховувати теорію розробки інтелектуальних систем навчання і навчальних сценаріїв. За кожне виконане завдання користувач отримує певні очки (бали), з накопичення яких будуть відкриватися нові рівні. В кожному наступному рівні буде збільшуватися складність завдань.

У додатку необхідно використовувати достатню кількість анімацій. Анімаційний пакет повинен являти собою анімацію деяких героїв казки, ані-

мований відгук на деякі дії користувача. Анімацію можна здійснити кількома способами: просте пересування об'єктів за допомогою коду; анімація за допомогою великої кількості картинок. Як вже зазначалося раніше, при першому запуску програми у користувача буде доступ до меню налаштувань, в яке користувач потрапляє, натиснувши на лівий верхній кут додатку. В даному меню користувач повинен мати можливість змінювати деякі параметри: включити або відключити звук; змінити мову інтерфейсу (англійська або російська). В рамках виконання магістерської роботи необхідно здійснити проектування та розробку інтерактивної системи навчання грі в шахи для дітей на платформі IOS. Розробка системи здійснюється використанням інтелектуальних навчальних ігрових сценаріїв для мобільних пристроїв iPad на платформі iOS. Для досягнення мети, необхідно виконати наступні завдання:

- провести дослідження та аналіз питань створення інтерактивної навчальної системи, проаналізувати існуючі класи навчальних систем і типи інтелектуальних навчальних сценаріїв;
- виконати проектування шаблонів сторінок і типових анімацій елементів системи у вигляді електронної книги;
- провести проектування сценаріїв навчальних фрагментів інтерактивної електронної книги;
- вибрати архітектуру та програмні засоби реалізації системи;.
- реалізувати анімаційні фрагменти і сторінок інтерактивної системи навчання, яка буде реалізована у вигляді електронної книги;
- розробити навчальні ігрові фрагменти для використання в електронній книзі.

Інтерактивну систему навчання необхідно реалізувати під платформу iOS 9.3.6 і вище для iPad. Для здійснення програної реалізації інтерактивної системи навчання для дітей, здійснивши дослідження та аналіз існуючих типів систем навчання, було обрано клас сценарно-орієнтованих навчальних систем – електронні книги.

3 ПРОЕКТУВАННЯ СТРУКТУРИ І СЦЕНАРІЇВ НАВЧАЛЬНИХ ЕПІЗОДІВ ІНТЕРАКТИВНОЇ СИСТЕМИ

3.1 Проектування шаблонів і інформаційного наповнення

Розробка інтерактивної системи навчання у вигляді електронної книги починається з інтеграції електронної версії книги в додаток. На сьогоднішній день існує два типи дозволу екранів. Для iPad, iPad 2 і iPad mini 3 дозволу екрану 1024x768 пікселів. Для iPad 4 і вище дозвіл екрана 2048x1536 пікселів. Оскільки хорошим тоном розробки додатків є універсальність додатки, необхідно мати два набори ресурсів. Якщо використовувати один набір ресурсів під певні моделі, то при імпортуванні на інший пристрій виникнуть проблеми з оптимізацією програми: для іншого типу пристроїв ресурси будуть стискатися або розтягуватися. Дана дія тягне за собою такі проблеми як зайве навантаження на пристрій під час відображення і втрата якості зображення після таких операцій. Виходячи з усього вищесказаного, можна зробити висновок, що для користувача більш важливим показником при завантаженні програми є не його обсяг, а поліпшена деталізація елементів програми, а також відсутність затримок при його відтворенні. З цього випливає, що треба розраховувати на два набори картинок. Завдяки відсутності будь-яких обмежень за кількістю ресурсів, було прийнято рішення переробити дизайн сторінок: в інтерактивній книзі, на відміну від надрукованої книги, можна приділити більшу увагу барвистості, анімації та інтерактивності. Оскільки дизайн сторінки планувалося переробляти, то був проведений аналіз можливих варіантів проектування шаблонів.

Самий оптимальний варіант – це реалізація кожної сторінки додатка окремо з накладеним текстом. Але, за попередніми розрахунками, розмір програми не задовольняв очікуванням. При наявності двох типів картинок під два екрани на двох мовах, кожна сторінка займала б від двох до п'яти мегабайт, а при наявності 100 сторінок, мінімальний розмір додатка становив би не менше 300 мегабайт. Не дивлячись на той фактор, що обсяг програми,

як зазначалося вище, не є основним показником, що впливає на вибір того чи іншого додатка, розмір програми більше 300 мегабайт – це не ефективний варіант реалізації сучасного додатку.

У зв'язку з цим, було прийнято рішення використовувати більш оптимізований підхід: розбити всі елементи книги на шари. Перший шар – це задній фон, кількість яких можна скоротити з 100 сторінок до 10. Після створення фону, на нього накладається шар з текстом. Залежно від вибраної мови, накладається певний мовний шар. Наступні шари містять наповнення сторінок всілякими картинками: саме це дозволить зробити кожну сторінку унікальною. Архітектура шаблону кожної сторінки інтерактивної системи навчання у вигляді електронної книги наведена на рис. 3.1.

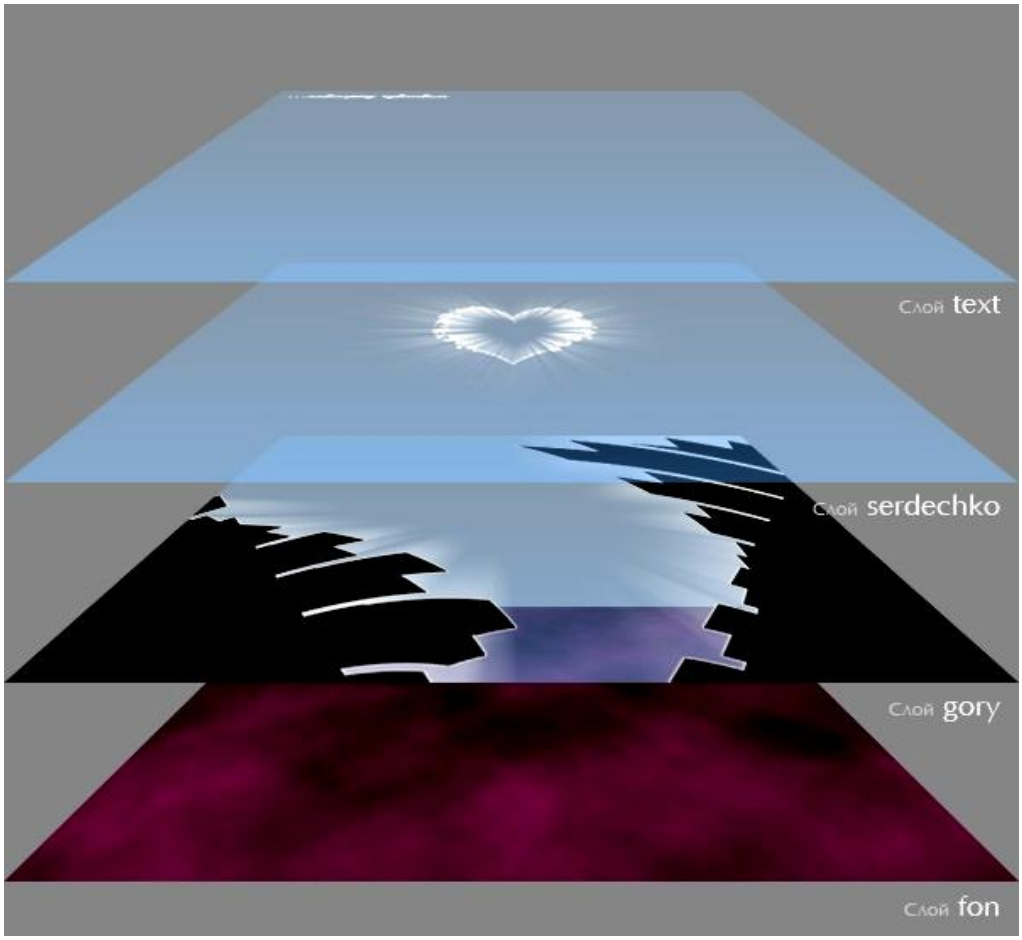

Рисунок 3.1 – Архітектура сторінки інтерактивної книги

З огляду на той факт, що цей додаток створюється як альтернатива друкованої книги, в додатку повинні бути присутні такі елементи як перегортання. Оскільки на кожній сторінці присутні інтерактивні елементи, то оптимальним рішенням є розташувати області перегортання в нижніх кутах. У правому нижньому куті – перегортання вперед, а в лівому нижньому – перегортання назад. Областю для переходу в меню обраний лівий верхній кут.

Цей додаток доцільно забезпечити ознайомлювальною частиною. Важливо також відзначити, що в опціях програми знаходяться такі можливості для користувача як зміна мови, відключення звуку.

3.2 Планування типових анімацій в розділах системи

Програна реалізація інтерактивної системи навчання для дітей, на підставі здійсненого дослідження та аналізу існуючих типів навчання та вибору класу системи навчання – електронні книги, передбачає вибір типу реалізації електронної книги. Існує три типи реалізації електронних книг для ОС iOS:

- класична електронна книга без будь-яких елементів інтерактивності, але система передбачає надання інтерактивних елементів навчання;
- окремий додаток, в якому можливості обмежуються можливостями розробника і вимогами компанії Apple;
- реалізація всієї книги за технологією Adobe Flash.

Основна відмінність інтерактивної книги для App Store і електронної книги для iBooks Author полягає в тому, що в інтерактивній книзі повинні підтримуватись як умова більше можливостей які не доступні в iBooks Author. До таких можливостей відноситься більш складна анімація (масштабування, 3D ефекти), прийняття користувачем участі в ігровому процесі і багато іншого що може бути придумано розробником.

Анімації бувають двох типів: стандартні і нестандартні. Стандартні анімації – це анімації, методи яких розроблені самою компанією Apple. Стандартні методи мають такі параметри як: координати, поворот, масштаб, рі-

вень прозорості, колір і інші. Не стандартні анімації реалізуються за допомогою інших бібліотек. Не стандартні анімації діляться на розширені стандартні (тобто мають більше можливостей), або покадрові анімації. Покадрова анімація представляє собою набір кадрів, які оновлюються з потрібною швидкістю за принципом кінострічки 80-х років. Покадрова анімація красивіша але енергоємна, якщо необхідно відображати кілька таких анімацій то пристрою постійно доводиться відкривати картинку і можливо застосовувати якесь масштабування [8][1)]. Для оптимізації цього методу використовують так званий атлас картинок (рис. 3.2).

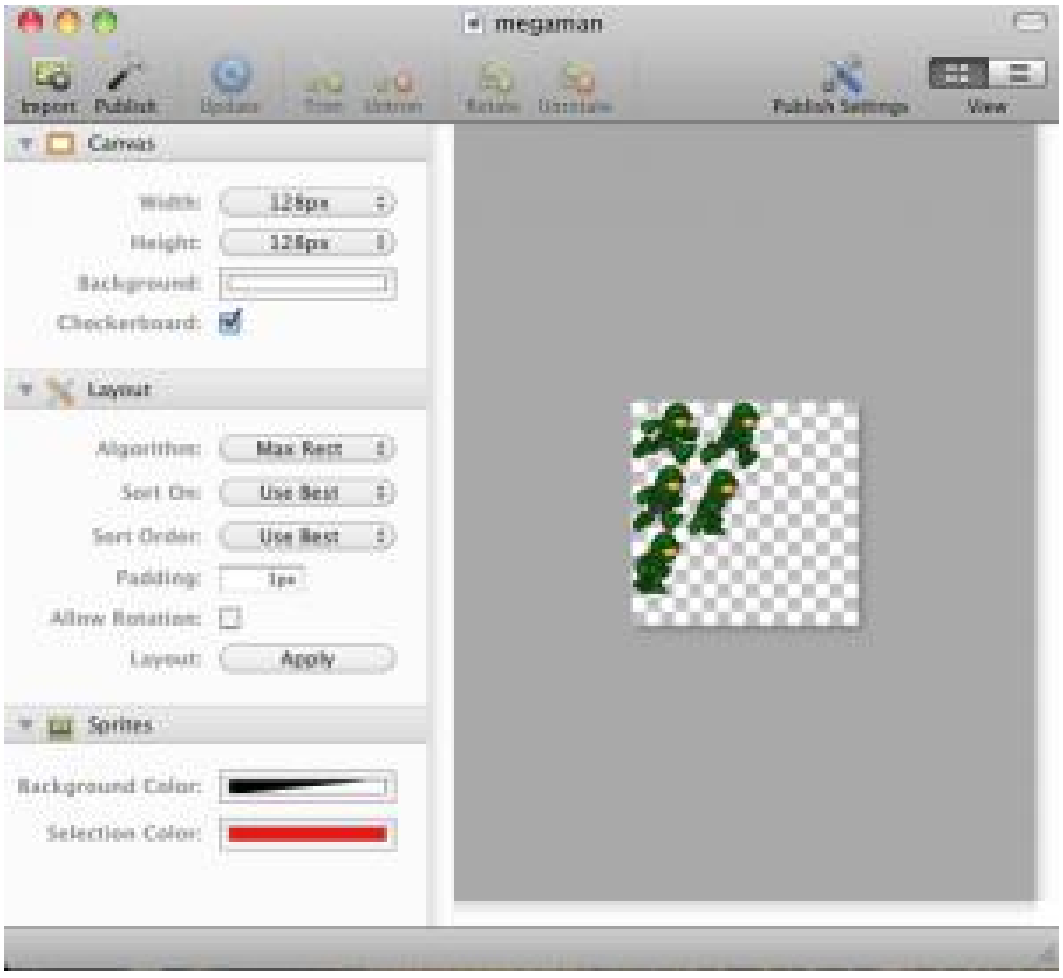

Рисунок 3.2 – Атлас картинки

[1)] [8] Разработка мобильных приложений: ИТ-Кластер. URL: http://itsiberia.ru/ru/services/details/service/mobile-solutions (дата звернення: 05.09.2019).

Всі картинки анімації поміщаються в одну картинку з роздільною здатністю 2048x2048 для пристроїв починаючи від iPhone 4s, iPad 2 та і iPad mini 1024x1024 для пристроїв нижче за класом. Одночасно створюється інший файл з координатами початку картинки в цьому файлі. У програмі дістаємо координати з файлу і по цих координатах поміщаємо картинку в масив і отримуємо масив картинок, які перебираємо під час анімації (рис. 3.3).

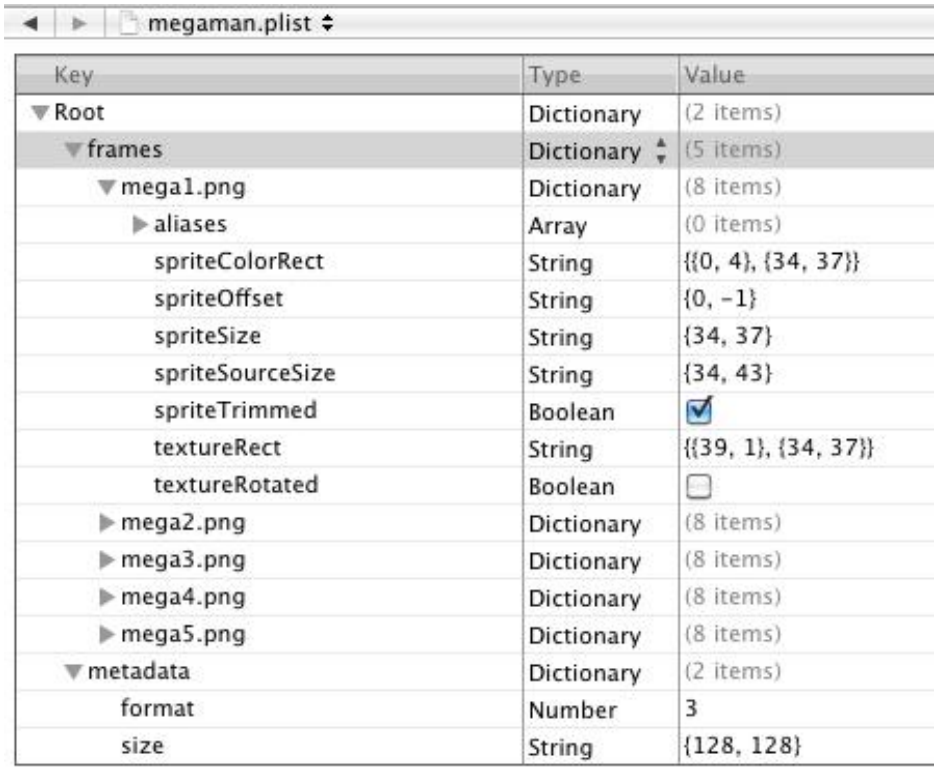

Рисунок 3.3 – Налаштування атласу картинок

Даний спосіб зручний у тих випадках коли є можливість помістити велику кількість картинок в один такий файл. Також існує технологія розробки з використанням Adobe AIR. Яка дає можливість створити єдине мобільний додаток, який може бути розгорнуто на безлічі мобільних пристроїв, включаючи смартфони та планшети під управлінням Android, iOS. Це досягається шляхом використання кросплатформових абстракцій (наприклад при доступі до фотографій), надання інформації про динамічних властивостей пристроїв (наприклад розмір екрану). У той же час AIR не заважає, коли допомога не

потрібна (наприклад при використанні API файлової системи). Розробка кросплатформових мобільних додатків вимагає від програміста знань про ефективне використання пам'яті і про життєвий цикл програми. Об'єднавши ці знання із середовищем виконання AIR, можна швидко створювати кросплатформові додатки. До недоліків зазначеної технології можливо віднести наступні: вся векторна анімація дуже гальмує; апаратне прискорення включити не має змоги [9][1)].

Перший спосіб найпростіший, для його реалізації людині не потрібні знання в програмуванні, досить завантажити спеціальний редактор для того щоб підготувати систему для в App Store. Другий спосіб набагато важче. Розробник реалізує всю логіку, перегортання і розробляє кожен елемент програми. Але даний спосіб набагато цікавіше для користувача тим більше якщо користувачем є дитина. Третій спосіб (розробка за технологією Adobe Flash) відрізняється від традиційних і менш цікавий.

3.3 Проектування сценаріїв навчальних фрагментів

У навчанні часто для закріплення нового матеріалу вводять практичні завдання. Цей додаток інтерактивної системи навчання грі в шахи для дітей на платформі iOS не є винятком і після кожного розділу необхідно реалізувати завдання з шахів. Ігрові елементи будуть доступні після кожного розділу і орієнтовані на той матеріал, який був наданий для вивчення в поточному розділі книги.

У додатку інтерактивної системи навчання грі в шахи для дітей на платформі iOS вводиться лічильник набраних очок, який відображає кількість набраних очок і відповідно рівень успішного проходження ігрового фрагменту. Планується в подальшому використання цього показника для визначення ходу розвитку ігрового сценарію в залежності від певних порогових значень

[1)] [9] О.С. Євсєєв. Комп'ютерна анімація: навчальний посібник для студентів. Х.: Вид. ХНЕУ ім. С. Кузнеця, 2014. 152 с.

набраних очок. Тобто якщо набрано малу кількість очок, сценарій повертає гравця в початок ігрового фрагмента. Якщо поріг набраних очок досить високий, здійснюється перехід до більш складної постановці ігрового сценарію. У цій реалізації будемо відображати тільки набрану кількість очок користувача.

3.3.1 Навчальний фрагмент «Вгадати клітинку»

Для повторення засвоєного матеріалу необхідно розробити додаток, в якому буде показана дошка з різними варіантами нотації. Користувачу необхідно натиснути на клітинку, яка з'явиться на екрані. Якщо користувач натиснув на правильну клітку, то він заробив очки, якщо ні – користувач оповіщається про те що натиснув не на правильну клітку. Даний фрагмент розроблений за методом програмного навчання Скінера, а саме, з використанням лінійної скинеровської послідовності. (рис.3.4)

Рисунок 3.4 – Алгоритм сценарію ігрового фрагменту «Вгадати клітинку»

Здійснено проектування послідовності дій сценарію ігрового фрагмента «Розставити фігури».

Крок 1 – користувач заходить в гру і вибирає рівень складності.

Крок 2 – з'являється дошка з нотацією або без неї.

Крок 3 – з'являється назва клітинки, яку необхідно натиснути.

Крок 4 – Користувач натискає на клітку: якщо натиснув вірно, то лічильник очок збільшується і з'являється назва нової клітини; якщо натиснув не вірно, то лічильник залишається без змін і позиція оновлюється.

Для здійснення програмної реалізації ігрового фрагменту інтерактивної навчальної системи грі в шахи для дітей необхідно навести алгоритм сценарію цього ігрового фрагмента «Вгадати клітинку» у вигляді блок-схеми для подальшої реалізації цього алгоритму у програмному коді .

3.3.2 Навчальний фрагмент «Падаючі фігури»

Для засвоєння вигляду видів фігур необхідно розробити навчальний фрагмент гри, яка орієнтована на зорове запам'ятовування вигляду кожної фігури в шахах. Даний фрагмент розроблений за типом скінеровської лінійної програми, а саме використаний у цьому фрагменті у вигляді тестування. На початку гри користувачу буде показана постать (вигляд фігури шахів), яку потрібно запам'ятати. Після цього фігури різних типів будуть падати зверху вниз. Завданням користувача є натиснути на необхідну фігуру. Після чого вона зникне і збільшиться лічильник очок.

Далі здійснено проектування послідовності дій сценарію ігрового фрагмента «Падаючі фігури », яка передбачає послідовне виконання користувачем наступних кроків.

Крок 1 – користувач запускає додаток.

Крок 2 – передається тип фігури яка буде використовуватися в додатку.

Крок 3 – дана фігура показується користувачу для запам'ятовування.

Крок 4 – випадковою послідовністю фігури падають зверху вниз.

Крок 5 – користувач натиснув на правильну фігуру. При натисканні на правильну фігуру лічильник збільшується і фігура зникає, а при натисканні на неправильно фігуру, лічильник не змінюється і збільшується швидкість падіння фігур. Блок-схема алгоритму сценарію ігрового фрагмента «Падаючі фігури» наведено на рис. 3.5.

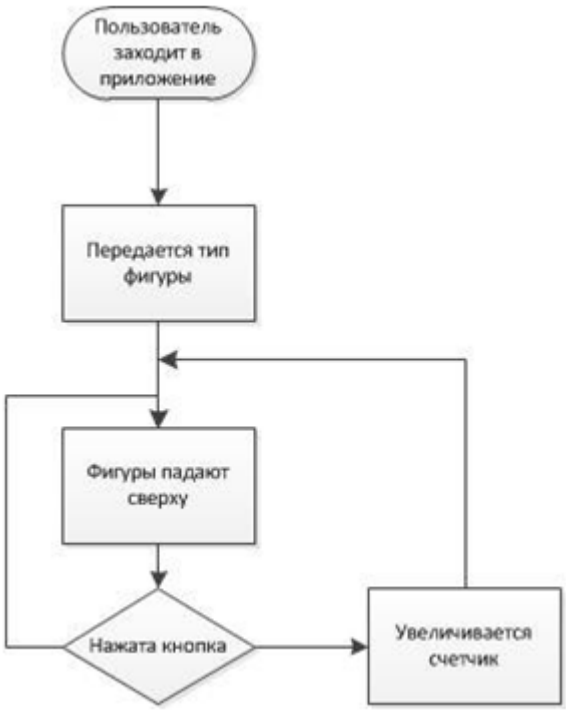

Рисунок 3.5 – Алгоритм сценарію ігрового фрагмента «Падаючі фігури»

Програмна реалізація сценарію зазначеного фрагменту в інтерактивній системі навчання для дітей здійснено за розробленим та наведеним алгоритмом.

3.3.3 Навчальний фрагмент «Збити всі цукерки»

Для запам'ятовування інформації про те, як ходять фігури, було вирішено реалізувати нескінченний цикл завдань за методом Краудера, в яких необхідно збити всі цукерки на дошці. Після того, як будуть збиті всі цукер-

ки буде збільшений лічильник очок і завдання оновиться автоматично і так до тих пір поки користувач не відчує, що він засвоїв даний матеріал і не перейде на наступну главу. Під час запуску додаток передається параметр – тип обраної фігури для якої необхідно відобразити завдання. Наприклад на малюнку інтерактивної системи навчання приведена шахівниця в початковому положенні ігрових фігур і цукерок з обраною типовою фігурою кінь. Графічне представлення алгоритму сценарію ігрового фрагменту «Збити всі цукерки» наведено на рис. 3.6.

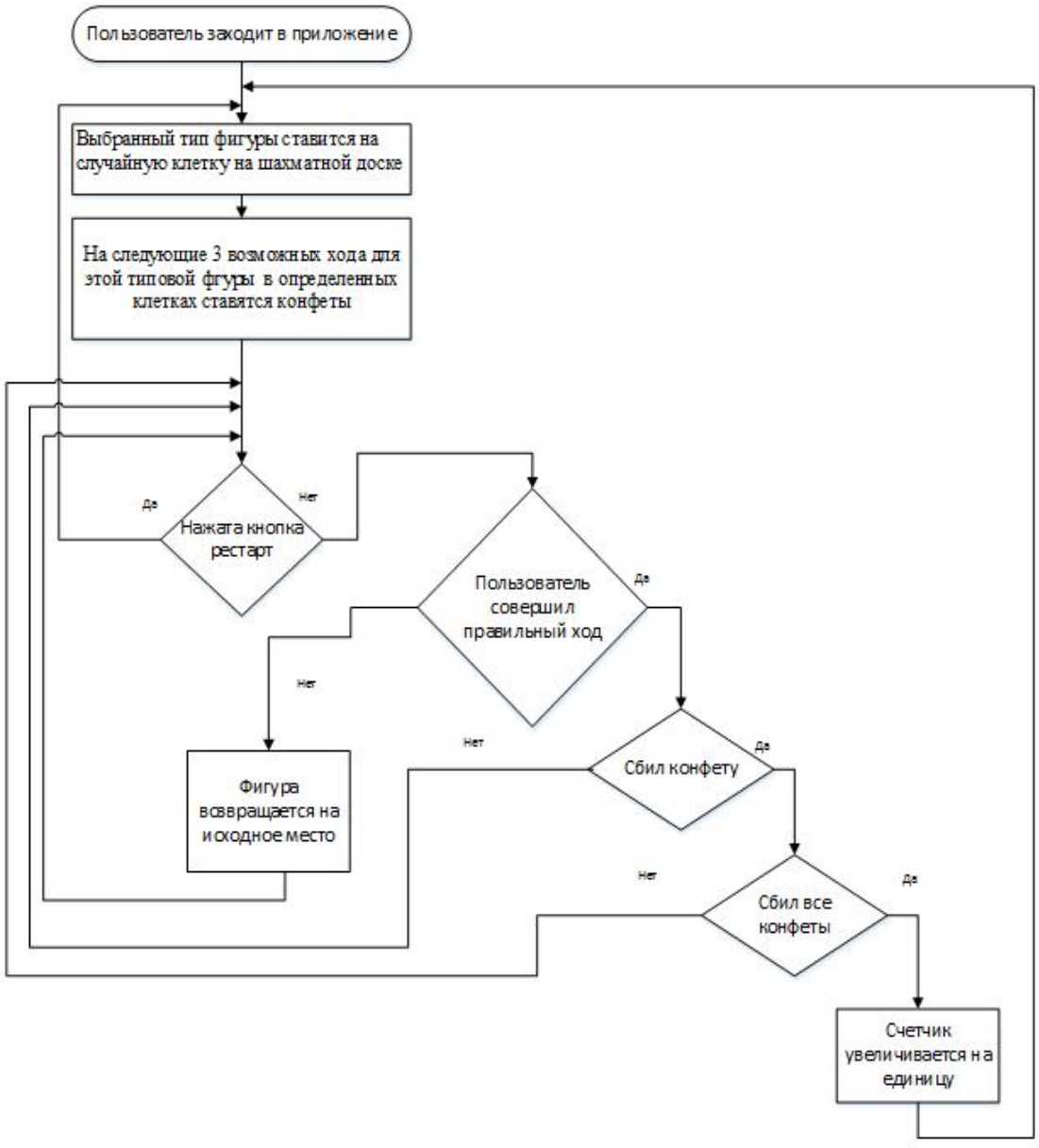

Рисунок 3.6 – Алгоритм сценарію ігрового фрагменту «Сбити всі цукерки»

Визначимо послідовність кроків сценарію ігрового фрагмента «Збити всі цукерки».

Крок 1 – користувач заходить в додаток.

Крок 2 – обраний тип фігури ставиться на випадкову клітинку на шахівниці.

Крок 3 – на наступні три можливих хода для цієї типової фігури в відповідних клітинках шахівниця розташовані цукерки.

Крок 4 – користувач здійснює хід. Якщо користувач здійснив неправильний хід, то фігура повернеться на вихідну позицію. Якщо користувач виконав правильний хід, але не збив цукерку, то програма зберігає цей хід і переходить в режим очікування. Якщо користувач збив цукерку, то здійснюється перехід на крок 4. Якщо користувач збив всі цукерки, то здійснюється перехід на крок 2 і лічильник очок збільшується на одиницю. Закінчення гри має здійснюватися після натискання кнопки вихід на будь-якому етапі гри.

3.3.4. Навчальний фрагмент «Гра на уважність»

Для тренування уважності необхідно розробити додаток, в якому буде показана дошка (шахівниця) з фігурами. Фігури протягом 10 секунд будуть поступово зникати. Завдання користувача полягати в тому, щоб поставити фігури на ті місця, на яких вони стояли до зникнення. Даний навчальний фрагмент також розроблений за методом скінеровской послідовності. Не залежно від результатів виконання цього завдання (вірно/не вірно), додаток запропонує наступне завдання. Якщо фігури розставлені правильно, то користувач отримає бал за виконане завдання і розташування фігур на шахівниці оновиться. Визначена послідовність дій проектованого сценарію ігрового фрагмента «Гра на уважність.

Крок 1 – користувач заходить в гру.

Крок 2 – випадкова кількість фігур розташовуються у випадкових клітинках дошки (шахівниці).

Крок 3 – протягом 10 секунд фігури поступово зникають.

Крок 4 – користувач розставляє фігури і натискає кнопку виконати. Якщо фігури розставлені правильно, лічильник очок збільшується і позиція оновлюється. Якщо фігури розставлені не вірно, лічильник залишається без змін, а позиція оновлюється. Алгоритм наведено на рис. 3.7

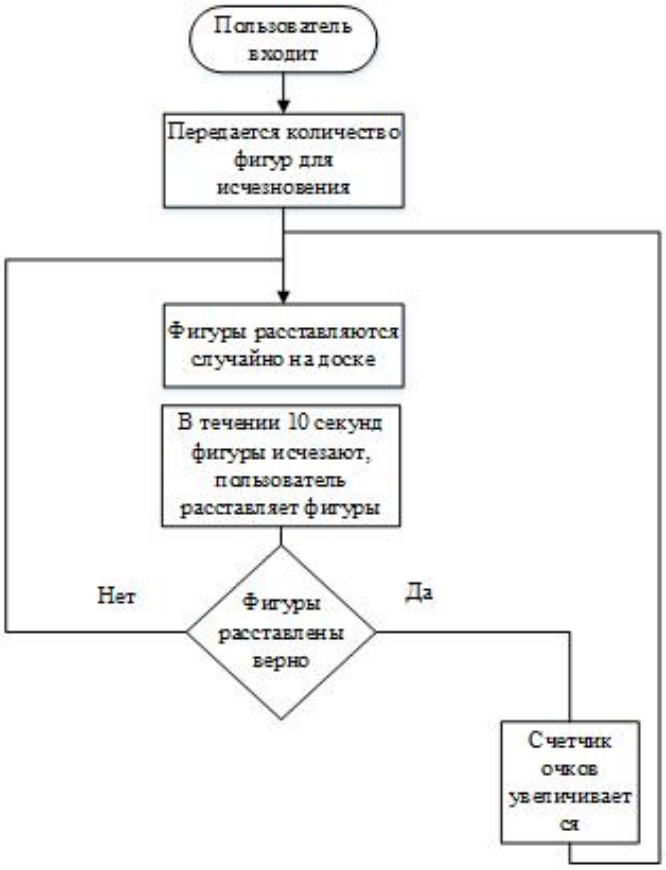

Рисунок 3.7 – Алгоритм сценарію ігрового фрагменту «Гра на уважність»

За розробленим алгоритмом необхідно подальше здійснення програмної реалізації ігрового навчального фрагменту для системи.

3.3.5 Навчальний фрагмент «Розташувати фігури»

Для повторення засвоєного матеріалу необхідно розробити додаток, в якому буде показана дошка з частково розставленими фігурами. Користувачу

необхідно розташувати фігури на шаховці у вірному порядку. Якщо користувач розставив фігури правильно то він заробив очки, якщо немає, то користувачу показуються клітини на яких фігури стоять неправильно і з'являється нове завдання. Даний фрагмент навчального сценарію розроблений за методом Краудера, тому присутня безліч правильних і не правильних відповідей які в сукупності складають єдину відповідь. Алгоритм сценарію ігрового фрагмента «Розставити фігури» наведено на рис. 3.8

Рисунок 3.8 – Алгоритм сценарію ігрового фрагмента «Розташувати фігури»

Визначимо послідовність дій сценарію ігрового фрагмента «Розташувати фігури»:

Крок 1 – користувач заходить в гру і обирає рівень складності.

Крок 2 – з'являється дошка з не розставленими деякими фігурами.

Крок 3 – користувач розставляє фігури і натискає кнопку виконати. Якщо фігури розставлені правильно, то лічильник очок збільшується і пози-

ція на дошці оновлюється. Якщо фігури розставлені не вірно, то лічильник залишається без змін і позиція оновлюється. За розробленим алгоритмом здійснена програмна реалізація наведеного сценарію у системі інтерактивного навчання дітей грі в шахи.

3.3.6 Навчальний фрагмент «Знайти фігури»

Для повторення засвоєного матеріалу необхідно розробити додаток, в якому користувачу буде показано безліч картинок і користувач повинен буде знайти картинки у вигляді шахових фігур. При натисканні на правильну картинку користувач буде отримувати окуляри. Даний фрагмент розроблений за методом скінеровського тестування що в даному випадку передбачає наявність тільки одної правильна відповідь. Алгоритм сценарію ігрового фрагмента «Знайти фігури» наведено на рис. 3.9.

Рисунок 3.9 – Алгоритм сценарію ігрового фрагмента «Знайти фігури»

Визначимо послідовність дій розробленого сценарію ігрового фрагмента «Знайти фігури».

Крок 1 – користувач заходить в гру, де відображається початковий фон.

Крок 2 – з'являються картинки поверх основного фону.

Крок 3 – користувач натискає на картинки які з’являються у системі. Якщо картинка є шаховою фігурою то користувач отримує очки Якщо картинка не є шаховою фігурою, то рахунок очок незмінний.

В даному розділі були визначені та спроектовані сценарії для навчальних фрагментів інтерактивної навчальної системи, яка здійснює навчання дітей грі в шахи. Згідно з зазначеними сценаріями буде здійснена програмна реалізація цих сценаріїв.

4 РОЗРОБКА ІНТЕРАКТИВНОЇ СИСТЕМИ НАВЧАННЯ

4.1 Вибір програмних засобів реалізації системи

На ринку програмного забезпечення існує велика кількість інтерактивних електронних книг. Практично у всіх реалізовані анімації, рух об'єктів, звуки і багато інших мультимедійних ефектів. Багато додатків реалізовано засобами фреймворку Cocos2d-x, який надає потужній та зручний інструмент розробнику. Cocos2d-x – це безкоштовний фреймворк з відкритим вихідним кодом. Дуже зручний для написання ігрових додатків, тому в ньому реалізований необхідний набір функцій на всі випадки життя. Для здійснення програмної реалізації інтерактивної системи навчання грі в шахи для дітей на платформі iOS були використані стандартні фреймворки Cocos2d-x та MediaPlayer.framework [10][1)].

Структура фреймворку MediaPlayer.framework наведена на рис. 4.1.

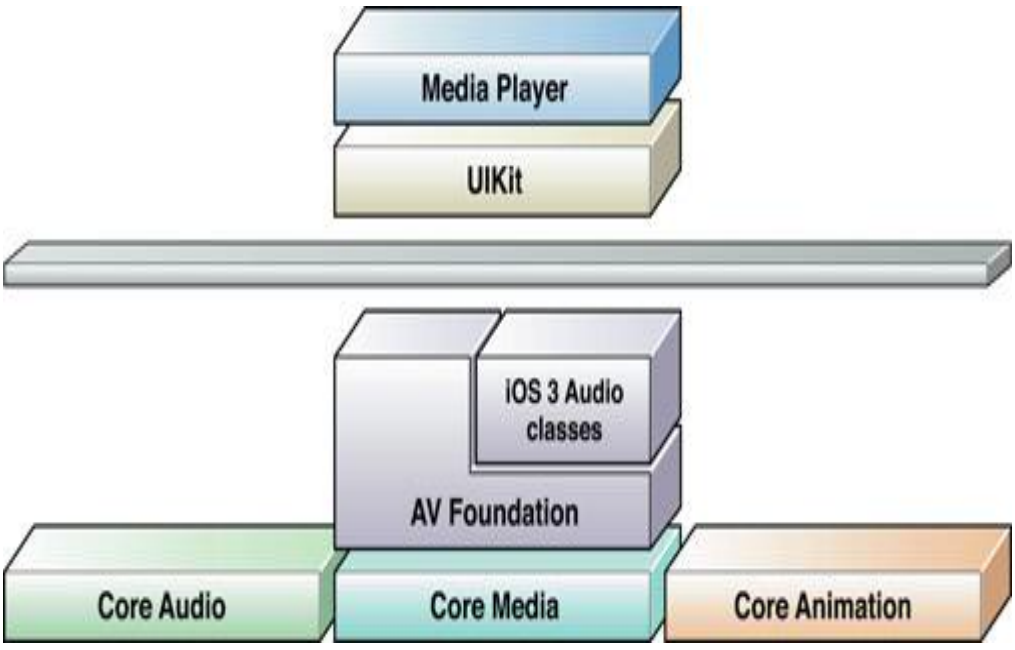

Рисунок 4.1 – Структура MediaPlayer.framework

[1)] [10] Cocos2d-x – разработка простой игры / Хабр. URL: https://habr.com/ru/post/270133/ (дата звернення: 15.10.2019).

Ієрархічна структура фреймворків для платформи iOS, що надають інструменти для роботи з відео має наступні бібліотеки [11][1)]:

– Media Player Framework – простий у використанні інструмент розробника, що підтримує більшість функцій «з коробки» та достатній для більшості додатків плеєр зі звичним та зручним для користувача інтерфейсом;
– AV Foundation Framework [13] – надає повний набір функцій для управління відтворенням відео і дозволяє реалізувати плеєр будь-якої складності: включає функції зйомки, редагування, запису і перекодування аудіо і відео;
– Core Media і Core Video Framework – описують низькорівневі типи даних і системні інтерфейси для маніпулювання медіа контентом.

Media Player Framework містить клас MPMoviePlayerController, який реалізує готовий до використання у власних додатках відеоплеєр, який практично повністю повторює плеєр стандартного додатка «Відео» (рис. 4.2).

Рисунок 4.2 – Вигляд відеоплеєру у Media Player Framework

У найпростішому варіанті можливо запустити плеєр буквально в кілька рядків коду, використовуючи MPMoviePlayerViewController.

[1)] [11] iPhone. Разработка приложений с открытым кодом. Пер. с англ. 2-е изд., перераб. и доп. СПб.: БХВ-Петербург, 2009. 368 с.

```
    MPMoviePlayerViewController * theMoviePlayer =
[[MPMoviePlayerViewController alloc] initWithContentURL: [NSURL
URLWithString: media_url]];
    [self presentMoviePlayerViewControllerAnimated:
theMoviePlayer];
```

В цьому випадку відбувається перехід на контролер плеєра, а інтерфейс вашого додатка стає недоступний користувачу до тих пір, поки користувач не закриєте плеєр. У трохи більш складному варіанті, екземпляр MPMoviePlayerController ініціалізується за умови посилання на контент, після чого його властивість view можливо додати в ієрархію поточного контролера. В цьому випадку плеєр точно також відображається в розгорнутому на весь екран вигляді, але тепер користувач може додати власні компоненти поверх нього. Користувач має можливість підписатися на сповіщення, що відправляються плеєром, і реагувати на них у власному коді при здійсненні користувачем наступних дій: початок відтворення, пауза, перемотування, початок і кінець відтворення через AirPlay, зміна режиму масштабування, перехід в повноекранний режим, зміни статусу завантаження відео, отримання метаінформації. Все ті ж дії можна виконувати і програмно через протокол MPMediaPlayback. Media Player дозволяє відтворювати контент у фоновому режимі, коли додаток згорнуто, або екран пристрою заблоковано. Для цього в налаштуваннях програми потрібно вказати параметр UIBackgroundModes зі значенням audio і задати категорію аудіо сесії AVAudioSessionCategoryPlayback, використовуючи API класу AVAudioSession.

Проблеми з цим фреймворком починаються, коли потрібно змінити функціональність плеєра, реалізувати будь-яку відсутню в стандартному плеєрі функцію, або змінити його візуальне оформлення. У таких випадках на допомогу розробнику приходить фреймворк AV Foundation [12][1)].

[1)] [12] Основы AVFoundation – Oceanize Geeks – Средний. URL: https://medium.com/oceanize-geeks/avfoundation-framework-de9a6a8ed453 (дата звернення: 20.10.2019).

Фреймворк AV Foundation не містить готових візуальних компонентів на зразок MPMoviePlayerController, але зате надає більш деталізований програмний інтерфейс для роботи з носіями і дозволяє реалізувати власний плеєр цілком з нуля. Перелічимо основні класи фреймворка, які використовуються для відтворення. AVAsset – базовий клас, що описує одиницю медіаконтенту, як єдине ціле, що включає треки, метаінформацію та інше. Важливою особливістю при роботі з цим класом є те, що він може бути не готовий до використання відразу після ініціалізації. Багато властивостей контенту, такі як тривалість, наявність треків, можливість відтворення, стають доступними асинхронно, а можуть і зовсім не бути отримані, наприклад, в разі недоступності контенту при завантаженні по мережі. Стан відтворення asset-а відокремлено в окремий клас AVPlayerItem, що дозволяє одночасно відтворювати один і той же asset незалежно один від одного в декількох плеєрах. Екземпляри класу AVPlayerItem відтворюються за допомогою об'єкта AVPlayer, перенаправляє висновок на спеціальний шар Core Animation, що має тип AVPlayerLayer [12][1)]. До цього шару можна застосувати всі ефекти оформлення, анімації, геометричні перетворення, як і до звичайних слоїв (рис. 4.3).

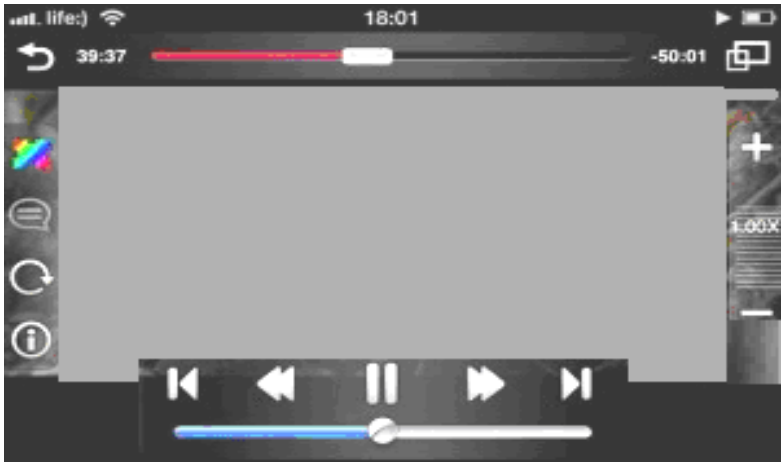

Рисунок 4.3 – Опції інтерфейсу класса AVPlayerLayer

[1)] [12] Основы AVFoundation – Oceanize Geeks – Средний. URL: https://medium.com/oceanize-geeks/avfoundation-framework-de9a6a8ed453 (дата звернення: 20.10.2019).

Крім базової функціональності відтворення, AV Foundation спільно з іншими фреймворками iOS надає багатий набір інших можливостей, що дозволяють виконувати з медіаконтентом практично всі. Перелічимо найбільш цікаві з них.

Клас AVAssetImageGenerator дозволяє незалежно від поточного відтворення витягувати з відео-ресурсу окремі кадри, послідовності кадрів або їх зменшені копії для цілей попереднього перегляду. Клас AVAssetReader дозволяє отримати найбільш повний контроль над відтворенням контенту. З його допомогою можна зчитувати і отримувати в зручному для додаткової обробки вигляді окремі семпли аудіо- та відеодоріжок asset-а. Наприклад, додати звукові ефекти, відобразити візуальне уявлення аудіотрека, додати різні фільтри до відео, натягнути відтворюється відео як текстуру на тривимірний об'єкт, і т.д. Можливості обмежуються фантазією розробника, а також швидкістю обробки кадрів, так як при роботі з AVAssetReader не може гарантуватися швидкість зчитування семплів в реальному часі. Об'єкт asset може містити кілька альтернативних аудіодорожек і субтитрів, наприклад для різних мов. У фреймворку AV Foundation для роботи з AirPlay передбачено властивість об'єкта AVPlayer usesAirPlayVideo WhileAirPlay ScreenIsActive, що дозволяє установкою одного прапора включити підтримку цієї технології. AirPlay може працювати не тільки для нативних додатків, але і в веб-середовищі, інтерфейс якого наведено на рис. 4.4.

Рисунок 4.4 – Реалізація технології AirPlay

Клас AVSynchronizedLayer дозволяє синхронізувати час в інших шарах додатка з часом відтворення контенту. Якщо користувач хочете, щоб швидкість анімації елементів інтерфейсу залежала від швидкості відтворення відео, або, щоб анімація відтворювалася в зворотному напрямку під час перемотування відео назад. Розглянемо можливості використання фреймворку QuartcCore. framework. На сьогоднішній день анімація в додатку є невід'ємною частиною, тому розробка сучасних додатків неможлива без застосування функцій анімації. Одна з потужних технологій платформи iOS – це Core Animation (CA). Розробникам дуже легко зробити прості речі, використовуючи цю структурою. Класів, що надаються Core Animation (QuartzCore.framework), досить багато. Основним класом є CAAnimation, але розробники ніколи не використовують його безпосередньо, так як він є абстрактним класом [12][1)]. Цей клас надає методи для створення анімації, дозволяє встановити атрибути анімації і забезпечує основну підтримку CAMediaTiming (рис. 4.5).

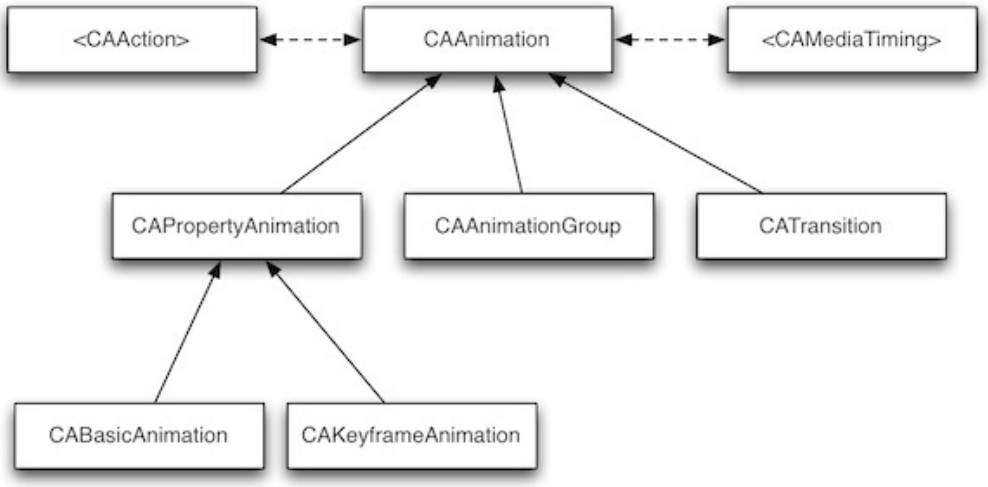

Рисунок 4.5 – Зв'язок між классами, які належать Core Animation

[1)] [12] Основы AVFoundation – Oceanize Geeks – Средний. URL: https://medium.com/oceanize-geeks/avfoundation-framework-de9a6a8ed453 (дата звернення: 20.10.2019).

Метою CA є не створення складних і заплутаних анімацій інтерфейсу, а замість цього, використання розробниками CA забезпечить поліпшення додатку та зробити його більш зрозумілим з точки зору зручності використання. CAAnimation має безліч підкласів: CABasicAnimation, CAKeyframeAnimation, CAPropertyAnimation і інші.

Суть методу анімації по ключовим кадрам полягає в поділі кадрів на ключові (keyframes) і проміжні (tweens) кадри. У ключових кадрах розробник задає властивості об'єкту: розмір, положення і т.д. У проміжних кадрах сама програма інтерполює властивості об'єкта, використовуючи додаткову інформацію, отриману від розробника, наприклад час анімації. Одна з найбільш сильних сторін сучасних мобільних пристроїв Apple – це графічна підсистема, що дозволяє писати якісні 2d і 3d ігри з відмінною продуктивністю і детальної графікою.

OpenGLES – це відкрита графічна бібліотека (Open Graphics Library), яка має свій API, свої змінні, і є самою найближчою точкою взаємодії між процесором і графічним чіпом (GPU). У пристроях на iOS є центральний процесор (CPU) і графічний процесор (GPU). GPU покликаний розвантажити центральний процесор від обробки графічних даних перед виведенням на екран. Іншими словами, OpenGL дозволяє розраховувати всі деталі кінцевого зображення на графічному чипі (графічний чип в рази швидше в розрахунках чисел з плаваючою точкою), а потужності центрального процесора залишаються для інших розрахунків, наприклад, ігрової логіки. Так само OpenGL надає безліч можливостей для зберігання інформації, даних і зображень в оптимальному для графічного чипа форматі для більш швидкої обробки. Ці дані будуть оброблятися безпосередньо графічним чипом. Логіка OpenGL дожу проста, і її можна пояснити трьома речами: примітиви; буфери; растеризація. Весь OpenGL працює навколо цих понять. Розглянемо їх детальніше. Примітиви в OpenGL складаються з трьох об'єктів: точка в просторі (x, y, z); лінія в просторі (складається з 2-х точок в просторі); трикутник в просторі (складається з 3-х точок в просторі). Точка може використовуватися як частка в про-

сторі. Лінія завжди суцільна і може використовуватися як вектор. Трикутник може бути однією з частин величезного полігону, який може складатися з тисяч трикутників. Буфери в OpenGL це тимчасове сховище. Вони діляться на 3 типи: frame buffer (кадровий буфер); render buffer (буфер що візуалізує); buffer object (буфер об'єктів) [13][1)].

Кадровий буфер є абстрактним з усіх трьох. Коли розробник формує підсумкове зображення в OpenGL, то можливо відправити його безпосередньо на екран або до кадрового буферу. Цей буфер зберігає інформацію про всі 3d об'єкти в просторі, віддаленості цих об'єктів в просторі, перекривання об'єктами один одного. Всі ці дані зберігаються в буфері у вигляді інформації про пікселі в двоїчним масиві.

Буфер візуалізації – це тимчасове сховище для одного зображення. Цей буфер являє собою колекцію кадрових буферів. Існує кілька видів візуалізації буфера: колір, глибина і шаблон. Буфер візуалізації кольору зберігає підсумкове кольорове зображення, створене рендером OpenGL. Це кольорове (RGB) зображення.

Буфер візуалізації глибини зберігає координату глибини (точка z) об'єктів в просторі. Буфер візуалізації шаблону зберігає видиму частину об'єкта (як маску частини, що є видимою). У буфері зберігається чорно-біле зображення. Буфер об'єктів зберігає інформацію про структурах і індексах, причому ця інформація на відміну від інших буферів може зберігатися протягом всього циклу виконання програми.

Структурою виступає масив, який описує 3d-модель як масив вершин, масив координат текстур або масив потрібних розробнику визначених даних. Індекси більш специфічні. Масив індексів використовується для вказівки, як сторони полігону що розробляється будуть побудовані на основі масиву структур. Розглянемо куб в просторі, який складається з 6 сторін і 8 вершин.

[1)] [13] Дэвид Вольф. OpenGL 4. Язык шейдеров. Книга рецептов. М.: ДМК Пресс, 2015. 462 с.

Кожна зі сторін це квадрат, але OpenGL вміє працювати тільки з трикутниками (рис. 4.6).

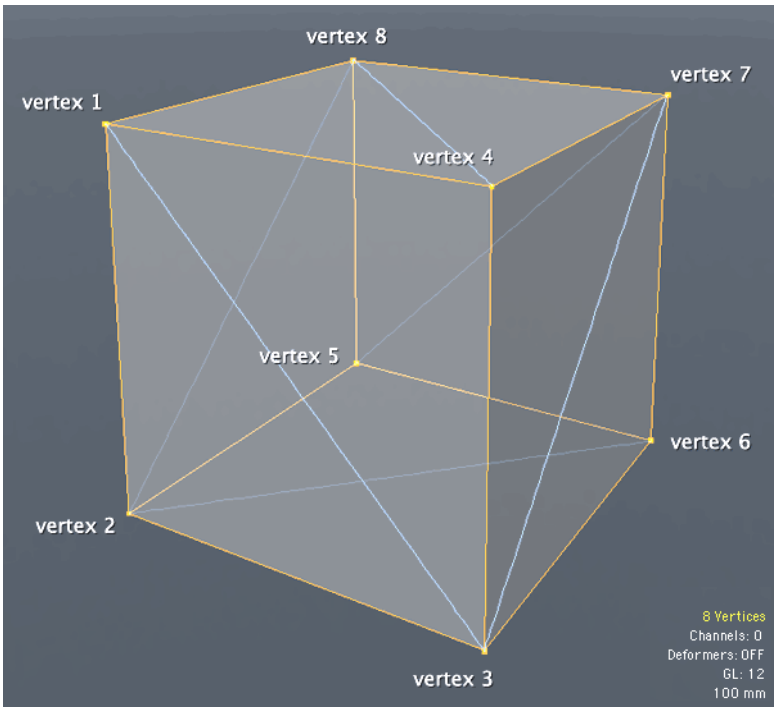

Рисунок 4.6 – Перетворення в OpenGL

Таким чином нам потрібно перетворити кожен квадрат в трикутники, що б відобразити його. Коли виконано перетворенн, то вийде, що 6 сторін перетворилися в 12. Трикутники в OpenGL є комбінацією з 3х вершин. Таким чином, що б «пояснити» OpenGL, що розробник хоче відобразити куб, нам буде потрібно намалювати 2 трикутника з наступними вершинами: {vertex1, vertex2, vertex3}, {vertex1, vertex3, vertex4}. Що б побудувати куб в OpenGL, потрібно створити масив з інформацією про 8 вершинах: {vertex 1, vertex 2, vertex 3, vertex 4, ...}, але замість того, що б кожен раз не перезаписувати цю інформацію для кожної сторони куба, можна побудувати масив індексів виду {0,1,2,0,2,3,2,6,3,2,5,6 ...}, де кожна комбінація з 3-х елементів (0,1,2 - 0 , 2,3 - 6,3,2) являє собою сторону трикутника. Ця можливість дозволяє один раз записати інформацію по вершинах в масив індексів і багаторазово її використ-

товувати. Таким чином, буфер об'єктів зберігає масиви структур {vertex1, vertex2, vertex3, ...} і масив індексів {0,1,2,0,2,3,2,6,3 ...}.

Величезною перевагою буфера об'єктів є те, що об'єкти оптимізовані для обробки на графічному чіпі і розробнику не потрібно зберігати масив в додатку після створення об'єкта в цьому буфері. Растеризация це процес, коли OpenGL збирає всю інформацію про 3d об'єктах (координати, вершини і т. Д.) І створює 2d зображення, як правило для відображення на екрані пристрою. У реалізації OpenGL Apple розробник не можете вивести зображення відразу на екран, а повинен обов'язково його помістити в кадровий буфер, а вже звідти, засобами EAGL (Extended Apple GL) вивести його на екран пристрою. У OpenGL ES використовується програмований конвеєр. Це означає що вся відповідальність за камери, освітлення, ефекти лежить повністю на розробнику. Робиться все це рейдерами [13][1)].

Шейдери, це невеликі частки коду, маленькі програмки, які виконуються безпосередньо на графічному чіпі для здійснення складних розрахунків. Фіксований конвеєр це повна протилежність програмованого. Фіксований конвеєр надає API для управління камерою, освітленням, ефектами, матеріалами. Для створення шейдеров в OpenGL ES використовується мова, дуже схожа на C, званий OpenGL ES Shader Language (GLSL ES або ESSL). Як же працюють рейдери. Створюють їх або в окремих файлах, або прямо в коді, головне що б рядок, що містить код шейдера, був відправлений в ядро OpenGL і скомпільоване там для використання. Шейдери працюють в парі – вершинні шейдери і фрагментні шейдери. Наприклад, у кубі вершинні шейдери (vertex shader), так само відомі як VS або VSH, це маленькі програми, які виконуються для кожної вершини. Якщо подивитися на куб на рисунку видно, що у куба виходить 8 вершин (5 вершина невидима). Відповідно, вершинний шейдер обробить цей куб 8 разів на графічному процесорі.

[1)] [13] Дэвид Вольф. OpenGL 4. Язык шейдеров. Книга рецептов. М.: ДМК Пресс, 2015. 462 с.

Вершинний шейдер задасть кінцеві позиції вершин з урахуванням положення камери, а так же підготує і виведе деякі змінні, необхідні для фрагментного шейдера. У OpenGL розробник не можемо ставити змінні безпосередньо для фрагментного шейдера, тільки через вершинний. Для розглянутого кубу вище, п’ята вершина невидима через розташування і поворот у куба в просторі, тому можливо бачити тільки 3 боки куба, і вони становлять 7 вершин. Це те, що робить фрагментний шейдер: він обробляє кожну видиму частину кінцевого зображення. Можна уявити кожну частину як піксель, але це не зовсім так, тому що піксеть в рендеринге OpenGL і в підсумковому зображенні, яке розробник бачить на екрані, може відрізнятися за розміром. Таким чином фрагмент може бути більше або менше ніж реальний піксель, в залежності від конфігурації пристрою і параметрів рендеринга. У кубі, наведеному вище, фрагментний шейдер обробить кожен піксель на всіх 3-х сторонах куба, сформованих за допомогою 7 вершин.

Усередині фрагментного шейдера можливо працювати з усім, що пов'язане з поверхністю: освітлення, тіні, відображення, текстури і будь-які ефекти. Результат роботи фрагментного шейдера – це колір пікселя в форматі RGBA (червоний, зелений, синій і альфа-канал). Вершинний і фрагментний шейдери працюють разом. Потрібно один вершинний шейдер і один фрагментний, що б гарантувати, що розробник не наробить помилок, а OpenGL завжди компілює пару VSH і FSH. Так як OpenGL це окремий API і виконується він в графічному чіпі, розробник не має прямого доступу до процесів, що відбуваються всередині, тому якщо виникає помилка всередині, то з додатком нічого не трапиться. Але як дізнатися, що якщо в одному з шейдеров є помилка або буфер рендеринга налаштований неправильно? Для таких випадків в OpenGL є спеціальний Error API. Він дуже простий і складається з декількох функцій, одна з яких це проста перевірка успішності завершення будь-яких операцій, що повертає yes або no. Таким чином, дуже просто швидко перевірити чи є помилки, і, якщо такі маю місце бути, то розробник отримаєте повідомлення з помилкою. Зазвичай перевірки розміщуються в

критичних точках додатка, наприклад при компіляції шейдеров, або при створенні буферів. При написанні майже будь-якого розважального додатка рано чи пізно виникає необхідність в програванні звукових ефектів. Багато з розробників йдуть по найпростішому і швидкому шляху, вибираючи для цього завдання стандартний AVAudioPlayer. Для деяких завдань його цілком вистачає. Але якщо розробляється гра або просто розважальне додаток, де активно використовуються звуки – його можливостей буде явно недостатньо. Цей клас не гарантує миттєве відтворення звуків в тому ж кадрі гри, в якому ви викликали функцію play. Також в ньому немає можливості контролювати пам'ять – завантажувати і вивантажувати ефекти тоді, коли вважаєте за необхідне. Є ще деякі завдання, які AVAudioPlayer не в змозі реалізувати – наприклад, просторове розташування джерел звуку, застосування до них різних ефектів і т.п

OpenAL – це кросплатфомовий програмний інтерфейс для роботи з аудіоданими. BiOS присутня його реалізація, але на жаль, це якраз один з небагатьох аспектів, який не досить добре документований Apple. У OpenAL існують три основні сутності [14][1)].

– Source, або джерело звуку. Біля джерела є свої координати в тривимірному просторі, змінюючи які можливо створювати різні просторові ефекти (більшість з яких, з огляду на особливості пристроїв на iOS, будуть помітні тільки в навушниках).
– Listener, або приймач звуку – це розробник (користувач) – сутність, яка слухає звук. В один момент часу може бути тільки один приймач. Як і у джерел, у нього є свої координати в тривимірному просторі.
– Buffer – контейнер, що містить незжатими звукову інформацію. Для того, щоб програти звук необхідно виконати наступні кроки: підготувати звукові файли в потрібному форматі; ініціалізувати OpenAL;

[1)] [14] OpenAL: что это за программа, возможности, исправление ошибок. URL: https://geekon.media/obzor-openal/ (дата звернення: 22.10.2019).

завантажити дані в буфер; створити джерело; зв'язати буфер з джерелом; програти джерело.

Тут перераховані лише ті можливості платформи iOS, які лежать на поверхні, але насправді їх набагато більше. Кількість різних комбінацій, які можливо скласти, використовуючи функції доступних фреймворків, безмежні. На жаль, поки що в iOS SDK існують і деякі помітні прогалини: відкриті API системи не включають коштів для живого мовлення медіапотоку, що знімається камерою і мікрофоном пристрою, в мережу в реальному часі. Ті програми, які це роблять, – Facetime, Skype, Ustream, –використовують закритий код, недоступний розробнику. Інший пробіл, особливо помітний при роботі з професійним відео, – відсутність повноцінної системи DRM, на зразок Adobe Access (в iOS вбудовано відтворення шифрованого контенту, а для управління ключами потрібно інтеграція сторонніх бібліотек – наприклад, Adobe Access або Widevine). Система FairPlay, за допомогою якої захищає контент сама Apple не є доступною для сторонніх розробників навіть на комерційній основі. Foundation Kit Framework – основна бібліотека, яка містить класи з префіксом NS. Середовище розробки Foundation – це найголовніше в наборі інструментів iOS-розробника. Воно надає кореневий клас NSObject, а також безліч фундаментальних блоків для iOS-розробки, від класів для чисел і рядків, до масивів і словників. Середовище розробки Foundation на перший погляд може здатися нудним, але воно пропонує неймовірну міць і її дійсно можна називати незамінною при розробці iOS-додатків. Відомо, що Cocoa Touch не включає AppKit, що надає класи і методи для роботи інтерфейсом в Mac OS X. Замість AppKit на платформі iOS для роботи з 2D використовується низькорівневий Сі-фреймворк Core Graphics, який також є частиною Mac OS X SDK , але використовується рідше, так як Сі-функції і робота з пам'яттю не так зручні і гнучкі, як класи Objective-C в AppKit. Крім того, оче-

видно, що при роботі з Core Graphics доводиться писати більше коду, який складніше розширювати і підтримувати [15][1)].

GameKit – це відкритий ігровий движок для платформ Windows, Mac OSX, Linux, Android і iPhone, який для програмування використовує мови C ++ і Lua. Движок інтегрований з багатьма сучасними бібліотеками для реалізації фізики і графіки. Особливості графіки:

– інтеграція з графічними движками Irrlicht і OGRE;
– використання графічної API OpenGL ES, а також DirectX;
– імпорт моделей формату FBX і Blend.

Особливості фізичного движка: в якості фізики використовується бібліотека Bullet, що дозволяє реалізовувати якісну фізику в іграх. Особливості звуку: звукова API – OpenAL, об'ємний звук.

Вимоги Apple до додатків: при перевірці додатків перед публікацією в App Store тестувальники Apple серед іншого перевіряють і функціональність програми, пов'язану з відеовещанієм. Перелічимо основні вимоги, які необхідно дотримати:

– відеовещання по каналах стільникового зв'язку має в обов'язковому порядку здійснюватимуться з використанням HTTP LS.
– аудіодоріжки у альтернативних відеопотоків з різним бітрейтом повинні бути повністю ідентичні, щоб уникнути звукових артефактів при перемиканні потоків.
– з тієї ж причини співвідношення сторін у всіх відеопотоків має збігатися. Один з альтернативних потоків повинен бути закодований з смугою 64 Кбіт/с і містити тільки аудіодоріжку.

4.2 Реалізація сторінок інтерактивної системи навчання

В ході розробки, з метою поліпшення додатку інтерактивної системи навчання для дітей на платформі iOS, та щоб уникнути проблем при подаль-

---

[1)] [15] Далримпл Марк, Киастер Скотт. Objective-C 2.0 и програмирование под Mac. Перевод с англ. М.: ООО «И.Д.Вильямс», 2010. 320 с.

шому редагуванні додатка, була розроблена структура шаблону. Створено файл plist (файл з параметрами), в якому збережена змінна, що відповідає за мову додатка і кілька масивів, що містять імена файлів, які містять шари заднього фону, картинок, англійського та російського тексту. Під час запуску додатка програма зчитує тип встановленого мови і зчитує вміст масивів. Масиви ставляться один до одного таким чином, що перша картинка заднього фону масиву background займає найнижчий рівень шарів. Потім накладається перша картинка з масиву pic в якому містяться індивідуальні елементи для кожної сторінки. І поверх цих шарів лягає шар з масивів rus або eng, в якому знаходиться картинка з текстом.

Один з інших варіантів реалізації – називати файли з картинками послідовними числами, наприклад 1.png 2.png, після цього в циклі перебирати ці файли і відображати в додатку. Але в процесі перебору файлів можна зіткнуться з певною проблемою. При зміні змісту, іноді буває необхідно вставити нову сторінку або прибрати існуючу. В цьому випадку доведеться перейменовувати всі файли масиву з великим індексом. Також, якщо є картинка яку необхідно вставляти на якусь сторінку, то доводиться дублювати картинку з новим ім'ям. У підсумку виходить багато картинок які в підсумку займають досить багато місця. При використанні першого методу можна створити файл з назвою empty.png і вставляти його в масив власноруч.

Метод незручний тим що необхідно вручну заповнити всі поля масиву. Якщо уявити, що книга на основі якої розроблюється інтерактивна система навчання грі в шахи для дітей, розрахована на 80 сторінок без додаткових елементів, то внести необхідно трохи більше 250 рядків в файл. Але є значний плюс у тому, що картинки можна називати як зручно розробнику.

При створенні програми були реалізовані наступні основні класи, які представлені у Додатку Б. Основними вважаються класи:

- BookData;
- TheBook;
- ThePage.

Клас BookData відповідає за перегортання сторінок. В даному класі існують такі методи як:

- turnPage – перегортання вперед. При здійснені виклику даного методу відбувається збільшення лічильника сторінок, з подальшим привласненням нових картинок на задній фон. Також відбувається перевірка на останню і мінус першу сторінку з подальшим перегортанням в початок або кінець;
- TurnBackPage – перегортання назад. При здійснені виклику даного методу відбувається зменшення лічильника сторінок, з подальшим привласненням нових картинок на задній фон. Також відбувається перевірка на останню і мінус першу сторінку з подальшим перегортанням в початок або кінець;
- returnTheCurrentPageString – метод повертає поточне ім'я картинки фону;
- returnTheCurrentPageTextString – метод повертає поточне ім'я картинки тексту;
- returnTheCurrentPagePictureString – метод повертає поточне ім'я картинки на якій знаходяться інші текстури сторінки;
- returnCurrentPageSpecialProperty – метод відповідає за виклик опцій для певних сторінок;
- gotoSpecificPage – метод визначає на якій сторінці буде запропоновано придбати книгу, якщо вона ще не придбана.

При ініціалізації клас BookData відкриває файл plist, заповнює необхідні масиви текстурами і привласнює змінним об'єкти масивів які знаходяться під номером аналогічним номером поточної сторінки. Також при ініціалізації відбувається перевірка куплена чи повна версія для обмеження кількості доступних сторінок.

У класі TheBook реалізована найперша сторінка. У даній сторінці реалізовано меню у вигляді хмар. При натисканні на хмару користувач переходить на певну главу. Перегортання меню відбувається при проведенні пальцем по

екрану. Внизу сторінки знаходяться кнопки для покупки програми і справа внизу перехід на першу сторінку. Клас ThePage за структурою аналогічний класу TheBook. У класі ThePage описуються всі дії, які відбуваються в книзі. Основні методи:

- buyTheBook – метод, який викликається для придбання платної версії;
- setUpIndexLinks – метод для відображення кнопок для переходу на сторінці змісту;
- update – метод, який при необхідності оновлює сторінку;
- ccTouchesBegan – метод для відстеження початкового натискання;
- ccTouchesMoved – метод для відстеження переміщення пальця по екрану;
- ccTouchesEnded – метод який по закінченню натискання виконує певні дії.
- setUpPurchaseLink – метод відображає кнопку для покупки книги;
- getRidOfFireball – метод відображає вбудовану анімацію;
- showChessBoard – метод, який відповідає за відображення шахівниці.

В даному класі при ініціалізації включається програвання аудіо файлу, після цього спрайтам присвоюються картинки сторінок, які визначені в класі BookData. Після цього йдуть перевірки на те, яку опцію повертає сторінка і забезпечує виконання певних дій.

4.3 Реалізація анімаційних фрагментів

При здійсненні програмної реалізації інтерактивної системи навчання грі в шахи для дітей на платформі iOS, реалізація анімаційних фрагментів здійснювалась двома типами: покадрова та програмна. Для реалізації покадрових анімацій був розроблений клас Anima-tionLayer. Для реалізації анімації необхідно передавати методу даного класу параметри анімації. До цих пара-

метрів відносяться: ім'я файлу, в якому знаходяться всі кадри, plist файлу, ім'я першої картинки, кількість кадрів, затримка, координати, масштаб.

Файл plist (рис.3.8) – це xml-файл, в якому знаходяться змінні, масиви і налаштування, які необхідно зберігати постійно і при потребі перезаписувати (рис. 4.7).

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple Computer//DTD PLIST 1.0//EN"
"http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
    <dict>
        <key>frames</key>
        <dict>
            <key>Gipo20001.png</key>
            <dict>
                <key>frame</key>
                <string>{{2,2},{284,422}}</string>
                <key>offset</key>
                <string>{-39,-32}</string>
                <key>rotated</key>
                <true/>
                <key>sourceColorRect</key>
                <string>{{94,71},{284,422}}</string>
                <key>sourceSize</key>
                <string>{550,500}</string>
            </dict>
            <key>Gipo200010.png</key>
            <dict>
                <key>frame</key>
                <string>{{248,534},{244,422}}</string>
                <key>offset</key>
                <string>{-23,-32}</string>
                <key>rotated</key>
                <false/>
                <key>sourceColorRect</key>
                <string>{{130,71},{244,422}}</string>
                <key>sourceSize</key>
                <string>{550,500}</string>
            </dict>
            <key>Gipo200011.png</key>
            <dict>
                <key>frame</key>
                <string>{{2,534},{244,422}}</string>
```

Рисунок 4.7 – XML-файл налаштування анімації Plist

У випадку з анімаціями, даний XML-файл зберігає координати елемента у файлі картинки, перегорнут він чи ні, масштаб та ім'я. (рис. 4.8).

Рисунок 4.8 – Файл з стисненням картинок

Кількість параметрів залежить від складності упаковки (стиснення). Чим більше і економічніше зображення упаковуються, тим більше параметрів необхідно відстежувати.

Програмна анімація використовувалася також для реалізації рухомих хмар і падаючих фігур. Для реалізації самої анімації необхідно задати об'єкту картинку, початкову позицію, параметри руху і розміри (рис.4.9).

Рисунок 4.9 – Програмна анімація рухомих хмар

4.4 Розробка ігрових навчальних фрагментів

При здійсненні програмної реалізації інтерактивної системи навчання для дітей грі в шахи на платформі iOS були використані розроблені шість алгоритмів навчаючих сценаріїв.

Розглянемо реалізацію навчальних фрагментів, які забезпечують у системі отримання і засвоєння навичок гри в шахи у дітей. Необхідно здійснити опис класів, розроблених для кожного фрагменту, і їх призначення. Також необхідно описати методи розроблених у фрагменті класів, які функції вони виконують.

4.4.1 Розробка навчального фрагмента «Вгадати клітку»

В даному ігровому фрагменті реалізовані такі класи як: SquareMemoryMenu, SquareMemory, BoardMemory.

У класі SquareMemoryMenu створюється меню для переходу в різні ігрові режими. Кожен пункт меню відповідає за перехід в ігровий режим в якому необхідно знаходити клітини певного типу, які показуються під час виконання фрагменту. Також є кнопка виходу з ігрового фрагмента назад в книгу.

Клас BoardMemory відповідає за елемент дошки, який чекає натискання на клітку, а також відображення елементів після позитивної або негативної відповіді. Основні методи, які реалізовані в даному класі:

- returnBulb – хмара з'являється після відповіді і підсвічування клітини червоним або зеленим кольором;
- ccTouchBegan – клас відповідає за натискання на клітку дошки;
- ccTouchEnd – клас відповідає за закінчення натискання на клітку.

У класі SquareMemory описуються клітини, які випадковим чином показуються під час виконання ігрового фрагмента. Також описуються ігрові режими. Всього існує три ігрових режиму. Наприклад, перший ігровий ре-

жим відображає дошку, в якій прописані всі клітини на дошці. Другий ігровий режим відображає дошку з лінійною натаціею дошки. Третій ігровий режим відображає дошку без нотації. Після кожної правильної відповіді додаються ігрові очки. Методи, реалізовані в даному класі:

– compaireYes – інкремент очок;
– compaireNo – дікремент очок;
– setBoard – вибір дошки відповідно до обраного рівня складності;
– setNewSquare – генерується випадкова клітина, яку потім необхідно натиснути на дошці.

Цей ігровий фрагмент інтерактивної системи навчання грі в шахи для дітей забезпечую запам’ятовування в ігровій формі розташування шахів.

4.4.2 Розробка навчального фрагменту «Падаючі фігури»

В інтерактивній системі навчання грі в шахи реалізовано згідно з розробленим алгоритмом сценарію ігровий «Падаючі фігури», який забезпечує у дітей розвиток умінь та навичок розпізнавання виду шахматних фігур (рис. 4.10).

Рисунок 4.10 – Навчаючий фрагмент «Падаючі фігури» системи

Для забезпечення цього ігрового фрагменту в системі були реалізовані наступні каласи: FigureControlMenu, FigureControl.

У класі FigureControlMenu створюється меню для переходу в різні ігрові режими. Кожен пункт меню відповідає за перехід в ігровий режим в якому необхідно ловити фігури певного типу, який вказаний в пункті меню. Також є кнопка виходу з ігрового фрагмента назад в книгу. У класі FigureControl описуються фігури, які беруть участь в даному ігровому фрагменті. Також описуються ігрові режими. Всього існує шість ігрових режимів. Наприклад, перший ігровий режим містить 6 параметрів. Перший параметр під назвою FigureDown містить тип фігури, яку необхідно ловити. Решта п'ять параметрів містять фігури, які також будуть падати, але після натискання на ці фігури очки будуть зменшуватися. Після успішного проходження завдання додаток зупиняється і показується анімація успішного завершення. В даному класі буди реалізовані наступні методи:

- count – інкремент очок;
- uncount – дікремент очок;
- gotoHelloWorldScene – вихід з ігрового фрагмента;
- gotoLoadScene – налаштування для першого пункту гри;
- gotoLoadScene2 – налаштування для другого пункту гри;
- gotoLoadScene3 – налаштування для третього пункту гри;
- gotoLoadScene4 – налаштування для четвертого пункту гри;
- gotoLoadScene5 – налаштування для п'ятого пункту гри;
- gotoLoadScene6 – налаштування для шостого пункту гри;
- init – ініціалізація класу;
- fire – вибір випадкової фігури для падіння;
- win – метод відповідає за дії після виграшу.

4.4.3 Розробка навчального фрагменту «Збити цукерки»

Для реалізації ігрового фрагменту створена модель шахівниці і гри в шахи в цілому – «движок», який може бути використаний в різних додатках

пов'язаних з представленням шахівниці і обробкою ігрових даних.. Для програмної реалізації гри були розроблені класи Figure і ChessBoard. У класі Figure були використані такі методи, як:

- initWithType: white: – метод, який ініціалізує тип фігури з заданим кольором; type – метод повертає тип фігури;
- isWhite – метод повертає значення типу Boolean, яка визначає колір фігури: біла чи ні (чорна);
- hasBeemMoved – метод привласнює змінної _move значення YES або NO. Необхідно для відстеження послідовності чергового ходу;
- possiblePlaces – метод, який обчислює масив клітин на які може ходити фігура;
- allPlaces – метод в якому реалізована логіка ходів фігур.

Застосування технології розташування шахових фігур в перспективі може вирішити велику кількість завдань. Для впровадження в майбутньому завдання різного роду був розроблений движок для гри в шахи. Наприклад, реалізація штучного інтелекту, або організація списку вже відомих шахових задач в яких розташування фігур буде зчитуватися з файлу (рис. 4.11).

Рисунок 4.11 – Навчальний фрагмент «Збити цукерки»

У класі ChessBoard відбувається парсинг рядка в якій задана позиція, розстановка распарсенного рядка і реалізація решти логіки, такої, як відстеження ходів, перетягування фігур і методи для реалізації гри в якій необхідно забрати пішаки, а саме розташування фігури на випадкової клітці і після цього розташування цукерок на наступних 3 ходу.

У створеному класі ChessBoard реалізовані наступні методи:

- makeNORMALString – в методі відбувається парсинг рядка;
- setupFiguresFromString – в методі відбувається розстановка фігур на дошці – шахівниці;
- randomFigurePlace – метод реалізований для ігрового елемента «Збити цукерки». В даному методі реалізовано випадкове розташування фігури на дошці і розташування цукерок які необхідно збити;
- gridWidth – повертає ширину клітини;
- gridHeight – повертає висоту клітини;
- figurePressDetected – в методі реалізовані всі дії, обчислення та виклику методів, які відбуваються після натискання на фігуру. Наприклад, визначення типу фігури, кольору, можливих ходів;
- tryToRstart – метод реалізований для поновлення позиції для ігрового фрагмента «Збити цукерки»;
- drowRect- – метод в якому реалізована шахівниця.

3.4.4 Розробка навчального фрагменту «Гра на уважність»

Для здійснення програмної реалізації ігрового фрагменту «Гра на уважність» були реалізовані два масиви. Перший масив заповнюється в той момент, коли програма випадково розставляє фігури. Другий масив заповнюється в момент, коли користувач розставив фігури. Для перевірки правильності рішення даного завдання порівнюються обидва масиви. Якщо вони ідентичні – тоді завдання вирішено вірно. Якщо хоч одна фігура не так поставлена – завдання вважається невиконаним.

В даному ігровому фрагменті реалізовані такі класи як: ChessArray, GameLayer, Rock, Bishop, Pown, Knight, King, Queen, CompareChessArray, Chess.

Клас ChessArray містить масиви розташування шахів на шахівниці для порівняння. Клас GameLayer відповідає за розстановку фігур навколо шахівниці. Класи Rock, Bishop, Pown, Knight, King, Queen класи містить інформацію про фігури. Клас CompareChessArray містить методи для порівняння масивів. Клас Chess відповідає за дії користувача над фігурами. Вигляд інтерфейсу навчального фрагменту «Гра на уважність» наведено на рис. 4.12.

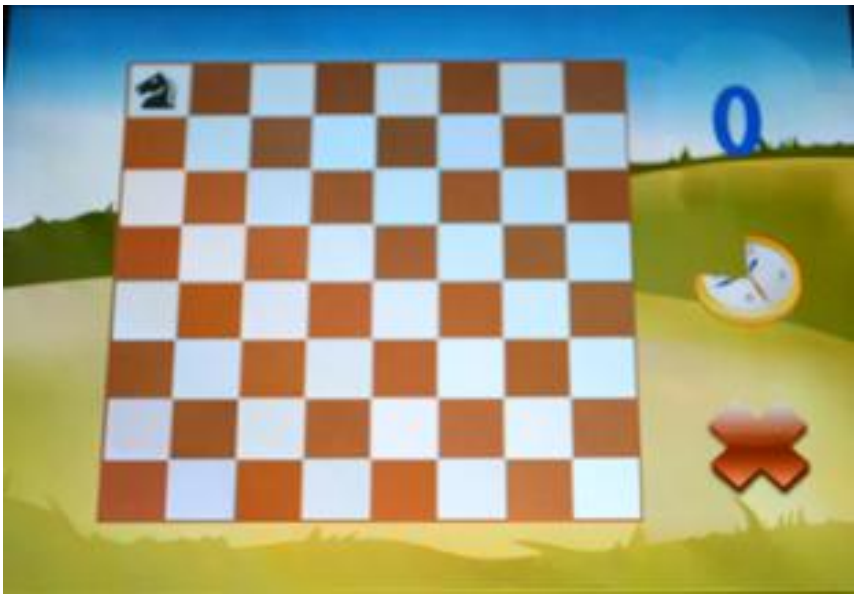

Рисунок 4.12 – Навчальний фрагмент «Гра на уважність»

Для користувача реалізована можливість запам’ятати розташування випадковим чином розставлених фігур на шахівниці, та розташувати їх при зникненні через деякий час на тому ж місці (у той же клітинки).

4.4.5 Розробка навчального фрагмента «Розставити фігури»

Для програмної реалізації навчального фрагменту «Розставити фігури» реалізовані наступні класи: StartPositionMenu, StartPosition.

У класі StartPositionMenu створюється меню для переходу в різні ігрові режими. Кожен пункт меню відповідає за перехід в ігровий режим в якому необхідно розставити різну кількість фігур певного типу. Кількість фігур, яку необхідно розставити зазначено в пунктах меню. Також є кнопка виходу з ігрового фрагмента назад в книгу.

У класі StartPosition створюється шахівниця, описуються фігури, які беруть участь і створюються масиви для реалізації даного ігрового фрагмента. Залежно від обраного ігрового режиму в файл передається число, яке відповідає за підрахунок кількості не розставлених фігур (рис.4.13).

Рисунок 4.13 – Навчальний фрагмент «Розставити фігури»

Для виконання всього функціоналу для даного навчального фрагменту були створені наступні методи класа:

– setBackground – присвоюється задній фон і рівень гри;
– setMenu – налаштування ігрового меню;
– setArray – створюється масившахівниці;
– compairArray – порівнюються масиви відповідей і завдання;
– setScore – налаштування рахунку очок;
– randomArray – налаштування масиву фігур які необхідно розставити;

– init – ініціалізація класу;
– setRandomPiece – ініціалізація фігур;
– setBoard – промальовування шахівниці.
– setChess – промальовування фігур.

Завдяки реалізації класів з переліченими методами в інтерактивній системі навчання для дітей на платформі iOS реалізовано навчальний фрагмент «Розставити фігури».

4.4.6 Розробка навчального фрагмента «Знайти фігури»

Для програмної реалізації навчального фрагменту «Знайти фігури» реалізовані наступні класи: FigureControlMenu, FigureControl.

У класі FigureControlMenu створюється меню для переходу в різні ігрові режими. Кожен пункт меню відповідає за перехід в ігровий режим в якому необхідно ловити фігури певного типу який вказаний в пункті меню. Також є кнопка виходу з ігрового фрагмента назад в книгу.

У класі FigureControl описуються фігури які беруть участь в даному ігровому фрагменті. Також описуються ігрові режими. Всього існує шість ігрових режимів. Наприклад, перший ігровий режим містить 6 параметрів. Перший параметр під назвою FigureDown містить тип фігури, яку необхідно ловити. Решта п'ять параметрів містять фігури, які також будуть падати, але після натискання на ці фігури очки будуть зменшуватися, т.к. відповіді-дії невірні. Після успішного проходження завдання додаток зупиняється і показується анімація успішного завершення. Перелічимо методи, які реалізовані в даному класі:

– count – інкремент очок;
– uncount – дікремент очок;
– gotoHelloWorldScene – вихід з ігрового фрагмента;
– gotoLoadScene – настройки для першого пункту гри;
– gotoLoadScene2 – настройки для другого пункту гри;

– gotoLoadScene3 – настройки для третього пункту гри;
– gotoLoadScene4 – настройки для четвертого пункту гри;
– gotoLoadScene5 – настройки для п'ятого пункту гри;
– gotoLoadScene6 – настройки для шостого пункту гри;
– init – ініціалізація класу;
– fire – вибір випадкової фігури для падіння;
– win – метод відповідає за дії після виграшу.

Таким чином, розроблена програмна реалізація інтерактивної системи навчання для дітей на платформі iOS забезпечує весь функціонал, що визначений у постановці завдання. Головна сторінка системи наведена на рис. 4.14

Рисунок 4.14 – Головна сторінка інтерактивної системи навчання

В системі реалізовано безліч сторінок з книги «Шахматы для малышей», що надають змогу використання для навчання дітей електронної книги, та шість навчаючих фрагментів у вигляді гри, що функціонують у додатку системи.

ВИСНОВКИ

В результаті виконання магістерської роботи здійснена розробка інтерактивної системи навчання грі в шахи для дітей з використанням інтелектуальних навчальних ігрових сценаріїв, що реалізовані для мобільних пристроїв iPad на платформі iOS. Відмінною особливістю реалізації інтерактивної системи навчання є розробка ігрових навчальних фрагментів, використання яких направлено на більш інтенсивне сприйняття інформації дітьми від трьох до семи років і проведення процесу навчання з використанням елементів і методів інтелектуальних навчальних систем.

При виконанні магістерської роботи, проведено дослідження існуючих класів інтелектуальних систем навчання; здійснено дослідження, аналіз та вибір типів сценаріїв навчання; проведено проектування сценаріїв та епізодів, типових анімацій для розділів інтерактивної системи навчання; здійснено вибір програмних засобів реалізації та виконана програмна реалізація фрагментів ігрового навчального середовища для мобільних пристроїв iPad на платформі iOS.

Програмна реалізація інтерактивної системи навчання грі в шахи для дітей виконана на платформі з iOS з використанням фреймворків Cocos2d-x та MediaPlayer.framework, відкритої графічної бібліотеки OpenGL. Класи і методи системи були реалізовані мовою програмування Objective-C. Розроблено ряд анімацій і багатошарових шаблонів сторінок з текстом і барвистими малюнками та шість навчальних ігрових моментів.

Застосування інтерактивної системи навчання грі в шахи для дітей надасть максимально бажаний ефект – засобами мобільного пристрою на платформі iOS, в ігровій формі дитина самостійно отримує навички гри в шахи. Регулярні заняття з освоєння гри в шахи навчать маленького шахіста розумно мислити, розсудливо і творчо підходити до вирішення найскладніших завдань.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Т.О. Піроженко, С.О. Ладивір, К.В. Карасьова. Дитина у сучасному соціопросторі: навчальний посібник. Кіровоград: Імекс-ЛТД, 2014. 272 с.
2. Новик І.М. Проектування навчальних компютерних ігор в освітньому процесі дошкільного навчального закладу – PSYH.KIEV.UA – Вісник психології і соціальної педагогіки. URL: https://www.psyh.kiev.ua/Новик_І.М._Проектування_навчальних_компютерних_ігор_в_освітньому_процесі_дошкільного_навчального_закладу (дата звернення: 15.09.2019).
3. Костина Ю.К. Шахматы URL: для малышей. Х.: Фактор, 2012. 96с.
4. И.А. Чмырь, М.Ф. Ус, А.В. Пискун. Интелектуальные системы обучения. Конспект лекций. Одесса: Издательский центр ОГАХ, 2000. 110 с.
5. В.Д. Шарко. Інтерактивні методи навчання: Досвід впровадження. Херсон: Олді-Плюс, 2000. 210 с.
6. Кашев С.С. Технология интерактивного обучения. Мн.: Белорусский верасень, 2005. 196 с.
7. Аверин В.А. Психология детей и подростков. СПб: Изд-во В.А. Михайлов, 1998. 379 с.
8. Разработка мобильных приложений: ИТ-Кластер. URL: http://itsiberia.ru/ru/services/details/service/mobile-solutions (дата звернення: 05.09.2019).
9. О.С.Євсєєв. Комп'ютерна анімація: навчальний посібник для студентів. Х. : Вид. ХНЕУ ім. С. Кузнеця, 2014. 152 с.
10. Cocos2d-x – разработка простой игры / Хабр. URL: https://habr.com/ru/post/270133/ (дата звернення: 15.10.2019).
11. iPhone. Разработка приложений с открытым кодом. Пер. с англ. 2-е изд., перераб. и доп. СПб.: БХВ-Петербург, 2009. 368 с.

12. Основы AVFoundation – Oceanize Geeks – Средний. URL: https://medium.com/oceanize-geeks/avfoundation-framework-de9a6a8ed453 (дата звернення: 20.10.2019).
13. Дэвид Вольф. OpenGL 4. Язык шейдеров. Книга рецептов. М.: ДМК Пресс, 2015. 462с.
14. OpenAL: что это за программа, возможности, исправление ошибок. URL: https://geekon.media/obzor-openal/ (дата звернення: 22.10.2019).
15. Далримпл Марк, Киастер Скотт. Objective-C 2.0 и програмирование под Mac. Перевод с англ. М.:ООО «И.Д.Вильямс», 2010. 320 с.