МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования

«Гомельский государственный технический университет

имени П.О.Сухого»

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

направление специальности 1-40 05 01-12 Информационные системы и технологии (в игровой индустрии)

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

к курсовому проекту

по дисциплине «Объектно-ориентированное программирование»

на тему: «Приложение, реализующее игру «Лабиринты» с иcпользованием Window Forms и графики OpenGL»

Исполнитель: студент группы ИТИ-21

Карамелева Д. А.

Руководитель: зав. кафедрой ИТ

Курочка К.С.

Дата проверки: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата допуска к защите: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата защиты: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Оценка работы: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Подписи членов комиссии

по защите курсового проекта: \_\_\_\_\_\_\_\_\_\_\_\_\_\_

Гомель 2021

Глава 1. Анализ необходимых компонентов для написания игры.

* 1. Основные этапы разработки игры

Разработка игры сложный и объемный процесс, требующий немало усилий. Написанием игры может заниматься как один человек, так и целая команда или корпорация. Однако, несмотря на количество человек, участвующих в процессе разработки, любая игра проходит несколько этапов:

1. идея
2. концепт
3. арт
4. программирование
5. звук
6. публикация
7. продвижение

Прежде всего необходимо определить цель игры. Необходимо определить будет ли эта игра заставлять задуматься над каким-то серьёзным вопросом или носить просто развлекательный характер. Главной целью игры может быть изучение иностранного языка. Большой популярностью пользуются развивающие игры для детей, помогающие малышам изучить не только компьютер, но и окружающий мир. Цель игры может заключаться в успокоении и расслаблении игрока, такие игры могут применяться в качестве терапии.

Первым шагом для разработки игры является составление концепции, которая должна включать в себя основную информацию об игре, такую как определение и анализ жанра, изучение тематики и целевой аудитории, краткое описание игры и её главных действий, экономику игры и социальные составляющие. Также стоит указать главные отличительные черты игры, чем она лучше конкурентов и что заставит игроков продолжить играть. Для наглядности нередко составляют специальную документацию. Концепт-документ позволяет правильно понять все задачи по разработке игры.

Далее создается графика игры. Во многих случаях от графики зависит восприятие игроком самой игры. Настроение, создаваемое в процессе игры, помогает игроку погрузиться в атмосферу и в некоторых случаях дает больше информации о происходящем в игре.

После изучения всех деталей игры и постановки необходимых задач, наступает этап программирования. Существует несколько способов написания игр. Самый простой способ – это использование движков. Для создание простой игры на движке достаточно просмотреть несколько обучающих видео в интернете. Основными современными движками являются CryEngine, Unreal Engine и Unity. [1] Наиболее распространённым из них является Unity, который рассчитан не только на 2D, но и на 3D графику.

Второй способ – использование шаблонов. Использование чужих участков кода или полное копирование и изменение компонентов на свои позволит легко и быстро создать игру. Однако, игра не только не будет оригинальной, но и также может иметь проблемы с законом об авторском праве.

Самый тяжелый и длительный по выполнению способ – написание кода вручную с нуля. Несмотря на всю сложность, процесс поможет лучше разобраться с языками, а также реализовать все идеи, не подгоняя игру под шаблоны. Для разработки игр без движка используются различные языки от Python и Pascal до C и Java. Выбор языка наиболее зависим от платформы, на которой будет реализована игра. Так, например, PHP и JavaScript хороши для написания нетяжелой браузерной игры, а для объемных и требовательных компьютерных игр больше подойдут С или C++. Кроссплатформенность языка Java позволяет легко адаптировать игры под любую операционную систему и создавать Android и веб-приложения. Язык Python пригодится для написания 2D и 3D-игр, простых квестов и RPG [1].

Следующим этапом в разработке игры является создание звука. Он, как и графика, призван погрузить игрока в атмосферу и помочь прочувствовать эмоции и чувства главного персонажа. Музыкальное сопровождение может предупреждать об опасности или задавать темп игре. Уникальный и особенный звук может стать визитной карточкой игры.

Для публикации игры существует множество различных сайтов, таких как itch.io, gamedev.ru, gamejolt. Также распространены специальные платформы, наиболее распространёнными из них являются Steam, EpicGames и Kartridge.

* 1. Анализ жанра компьютерной игры Лабиринты.

В настоящее время существует множество различных жанров игр, наиболее распространенными из которых являются шутер, боевая арена и бродилка [2]. Также могут одновременно комбинироваться несколько жанров. В данной работе рассматривается игра лабиринт как подвид жанра головоломка с добавлением элементов жанра шутер.

Классификация лабиринтов насчитывает десятки видов.

Лабиринты подразделяются на несколько типов по количеству разветвлений. Первый тип – без разветвлений. У игрока есть только один путь, и он не может заблудится. Во втором типе лабиринта присутствуют несколько комбинированных дорожек. Игроку нужно продумывать свой дальнейший путь на каждой развилке. Для третьего типа характерна комбинация разветвлений и прямых путей [3]. В некоторых играх игрок сам может менять форму лабиринта, открывая новые проходы.

Лабиринты имеют различный угол обзора: сверху, сбоку и от первого лица. Главной задачей может являться прохождение лабиринта или победа над врагами. В некоторые игры лабиринты могут быть вставлены как часть игры. Также игра может состоять из множества небольших лабиринтов-уровней или быть одним масштабным лабиринтом.

Для разнообразия в игру часто вставляют различные механики. Телепорты, лазерные пушки, всевозможные барьеры, механические двери и другие необычные идеи, которые могут быть привлекательными для игроков.

Игры-лабиринты пользуются большей популярностью среди детей. Они развивают логику и мышление в игровой форме, что делает их не только интересными, но и полезными.

* 1. Особенности языка С#

C# – современный объектно-ориентированный язык программирования. Является быстро развивающимся и достаточно мощным для разработки и поддержки даже крупных веб-проектов, обслуживающих ежедневно миллионы пользователей. C# принадлежит семейству С-подобных языков. Его синтаксис и принципы работы схожи с такими языками как C++, Java и JavaScript.

Синтаксические конструкции C# унаследованы не только от C++, но и от Visual Basic. Так, например, в C#, как и в Visual Basic, используются свойства классов. Как C++, C# позволяет производить перегрузку операторов для созданных пользователем типов. При этом C# синтаксически не менее чист, чем Java, так же прост, как Visual Basic, и обладает практически той же мощью и гибкостью, что и C++. Главной особенностью является то, что в данном языке предусмотрена полная поддержка использования программных интерфейсов. C# реализован на платформе .NET, ядро которой работает на уровне общего языка, известного под названием Common Language Subset (CLS, также называется Common Language Specification), и обеспечивает связь между всеми языками и библиотекой классов [4]. CLS определяет минимальный набор функций, которые должны поддерживаться всеми языками.

Платформа .NET позволяет передавать объекты (ссылки или значения) через границы программных модулей. Также предусмотрена полная поддержка аспектно-ориентированных программных технологий, таких как атрибуты.

По аналогии с Java, C# не поддерживает указатели, использует встроенных механизм сборки мусора, что позволяет программисту не задействовать деструкторы, и освобождает от необходимости отслеживать процесс выделения памяти [4].

Мощная библиотека каркаса поддерживает удобство построения различных типов приложений на C#, позволяя легко строить Web-службы, другие виды компонентов, достаточно просто сохранять и получать информацию из базы данных и других хранилищ данных. Реализация, сочетающая построение надежного и эффективного кода, является немаловажным фактором, способствующим успеху C#.

Так как язык принадлежит компании Microsoft, то применяется практически во всех продуктах, которые были разработаны или куплены компанией. Наиболее интересным из них является кроссплатформенный игровой движок Unity, позволяющий создавать двухмерные и трехмерные игры. Большей популярностью пользуется у инди-разработчиков из-за своей простоты.

* 1. Window Forms как основа для графического приложения

В современных компьютерных и мобильных играх графический интерфейс является неотъемлемой частью. Для его реализации с помощью платформы .NET применяются разные технологии: Window Forms, WPF и другие.

В данной работе графическое приложение будет реализовано с помощью Window Forms. Это графический пользовательский интерфейс для приложений .Net Framework, имеющий набор управляемых библиотек и предлагающий обширную клиентскую библиотеку для предоставления доступа к элементам графического интерфейса из управляемого кода. Приложение Window Forms предоставляет оболочку, состоящую из набора классов C ++ для разработки приложений Windows, и каждый элемент управления в приложении формы Windows является конкретным экземпляром класса. Он предоставляет различные элементы управления, такие как текстовые поля, кнопки, метки и веб-страницы, а также параметры для создания настраиваемого элемента управления. Для этого в Visual Studio предусмотрен инструмент конструктора форм Windows, который обрабатывает элементы управления в форме и упорядочивает их в соответствии с желаемым макетом для добавления кода для обработки событий [5].

Window Forms, или формы, являются более старым фреймворком, чем WPF. В следствии этого можно определить основные различия. В первую очередь Window Forms привлекают своей простотой и понятностью в изучении. Поскольку технология выпущена давно, то является хорошо испытанной и протестированной.

В отличии от WPF, формы не имеют огромного количества различных элементов, требующих специфических сложных настроек. Несмотря на это, в них присутствуют более 60 элементов с индивидуальными настройками. К сожалению, Window Forms не имеет такой гибкости в настройке как WPF, в связи с чем проекты, написанные на данном фреймворке, выглядят куда более просто и незамысловато, чем на WPF.

Window Forms были созданы для работы с определенными сценариями, поэтому создать различные элементы не составляет никакой трудности. С другой стороны, WPF не ограничена жесткими рамками как Window Forms, что позволяет расширить функционал и индивидуализировать интерфейс. Например, немалых усилий потребует создание ListView с изображением и хорошо выровненным текстом, тогда как в Window Forms это всего лишь одна строка кода [5].

С точки зрения написания, дизайнер Visual Studio лучше приспособлен к Window Forms, так как в WPF больше необходимо делать самому.

В следствии перечисленных выше различий, можно сделать вывод, что Window Forms больше подойдет для знакомства с разработкой пользовательского интерфейса, а WPF уже для более продвинутого изучения.

* 1. Графический интерфейс приложения с помощью OpenGL

Для программирования графики компьютерных игр используются специальные интерфейсы, которые также находят применение не только в играх. Самые известные из них это — библиотеки DirectX и OpenGL.

OpenGL, или открытая графическая библиотека, это популярный графический интерфейс (API) для разработки приложений с использованием 2D и 3D графики. Он включает в себя около 150 различных команд, с помощью которых можно определять различные объекты и производить рендеринг. Пользователь задаёт их местоположение в трёхмерном пространстве, определяет другие параметры (поворот, растяжение), задаёт свойства объектов (цвет, текстура, материал), положение наблюдателя, а библиотека OpenGL позаботится о том, чтобы отобразить всё это на экране. Поэтому можно сказать, что библиотека OpenGL является только воспроизводящей (Rendering) библиотекой, потому что она не поддерживает какие-либо периферийные устройства, таких как клавиатура и мышь, она также не поддерживает никаких менеджеров окон. Программист должен сам заботится том, как обеспечить взаимодействие периферийных устройств с библиотекой OpenGL [6].

OpenGL имеет хорошо продуманную внутреннюю структуру и довольно простой процедурный интерфейс. Несмотря на это с помощью OpenGL можно создавать сложные и мощные программные комплексы, затрачивая при этом минимальное время по сравнению с другими графическими библиотеками [7].

Главным преимуществом библиотеки OpenGL является то, что она не зависит то какой-нибудь операционной системы, как например, DirectX. Кроссплатформенность позволяет легко перенести написанный под Windows код программы на Linux или другие ОС. OpenGL отличается стабильностью, а значит все выпускаемые дополнения реализуются таким образом, чтобы сохранить совместимость с более старым программным обеспечением.

В данной работе для реализации графики приложения будет использоваться библиотека OpenTK.

OpenTK – это низкоуровневая библиотека, которая является обёрткой для OpenGL, OpenGL ES, OpenCL и OpenAL. Данная библиотека предназначена для игр, научных приложений, а также для любых других приложений, которым требуется графика, аудио и вычислительная функциональность. OpenTK предоставляет несколько служебных библиотек, включая пакет математической и линейной алгебры, систему управления окнами и обработку ввода. Библиотека является кроссплатформенной и работает как на Mono, так и на .NET.

* 1. Шаблоны проектирования

Шаблоны проектирования – это допускающие многократное использование оптимизированные решения проблем программирования, с которыми сталкиваются программисты [9]. Они служат для упрощения работы и предотвращения решения одних и тех же задач путем написания длинного и сложного кода. Другими словами, это некоторый шаблон, независящий от языка, который должен быть реализован в надлежащей ситуации. Любой шаблон проектирования следует применять с большой осторожностью, так как при реализации его в ненадлежащем месте его действие может быть разрушительным и принести много проблем.

Шаблоны проектирования по своей сути являются хорошо продуманными решениями проблем программирования. Многие программисты уже сталкивались с этими проблемами ранее и использовали для преодоления эти «решения». Более того, уже созданные шаблоны хорошо проверены и протестированы. В связи с этим, при необходимости решения такой же задачи есть возможность воспользоваться чужим опытом, облегчив и ускорив работу.

В данной работе предусмотрено использование двух шаблонов проектирования: Декоратор, или оболочка, и Фабричный метод.

Декоратор – это структурный паттерн проектирования, который позволяет динамически добавлять объектам новую функциональность, оборачивая их в оболочки. Шаблон Декоратор предоставляет гибкую альтернативу практике создания подклассов с целью расширения функциональности.

Класс ConcreteComponent — класс, в который с помощью шаблона Декоратор добавляется новая функциональность. В некоторых случаях базовая функциональность предоставляется классами, производными от класса ConcreteComponent. В подобных случаях класс ConcreteComponent является уже не конкретным, а абстрактным. Абстрактный класс Component определяет интерфейс для использования всех этих классов. Добавляемая функциональность реализуется в небольших объектах. Преимущество состоит в возможности динамически добавлять эту функциональность до или после основной функциональности объекта ConcreteComponent.

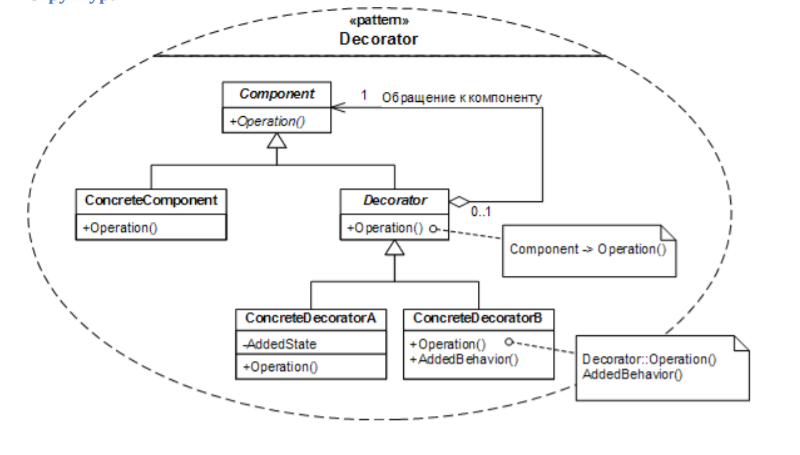


Рисунок 1 – Структура паттерна Декоратор

Фабричный метод – порождающий шаблон проектирования, предоставляющий подклассам интерфейс для создания экземпляров некоторого класса. В момент создания наследники могут определить, какой класс создавать. Это означает, что Фабрика делегирует создание объектов наследникам родительского класса. Это позволяет использовать в коде программы не специфические классы, а манипулировать абстрактными объектами на более высоком уровне.

Абстрактный класс "Creator" объявляет Фабричный Метод, возвращающий объект типа "Product" (абстрактный класс, определяющий интерфейс объектов, создаваемых фабричным методом). "Creator также может определить реализацию по умолчанию Фабричного Метода, который возвращает "ConcreteProduct". " ConcreteCreator" замещает Фабричный Метод, возвращающий объект " ConcreteProduct ". "Creator" "полагается" на свои подклассы в определении Фабричного Метода, возвращающего объект " ConcreteProduct ".

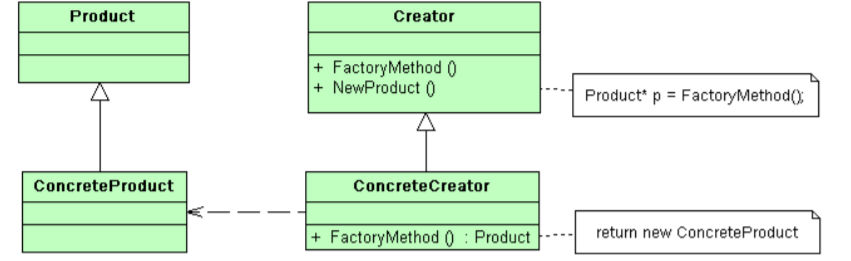


Рисунок 2 – Структура паттерна Фабричный метод

Главной особенностью данного метода является то, что он избавляет проектировщика от необходимости встраивать в код зависящие от приложения классы, а также позволяет установить связь между параллельными иерархиями классов.

Список использованных материалов

1. https://skillbox.ru/media/gamedev/kak\_programmirovat\_igry/
2. Индустрия компьютерных игр автор: Седых И. А.
3. Савард Дж. Лабиринты. – М.: Ниола 21 век, 2005
4. Герман О.В., Герман Ю.О. Программирование на Java и C# для студента
5. https://www.wpf-tutorial.com/ru/2/несколько-слов-о-wpf/wpf-vs-winforms/
6. OpenGL. Программирование компьютерной графики.Автор: Френсис Хилл, 2008
7. Учебное пособие OpenGL. компьютерная графика. автор Васильев С.А.
8. https://opentk.net/faq.html
9. Эрик Фримен, Элизабет Фримен, Кэтти Сьерра, Берт Бейтс — Паттерны проектирования, 2011
10. Design Patterns with examples in C#, D? 2012