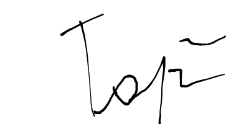
|  |  |  |
| --- | --- | --- |
|  |  |  |
| МИНОБРНАУКИ РОССИИ | | |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** | | |
| Институт искусственного интеллекта | | |
| Кафедра программного обеспечения систем радиоэлектронной аппаратуры | | |

КУРСОВАЯ РАБОТА

по дисциплине «Методы и стандарты программирования»

на тему: «Создание компьютерной игры Змейка»



|  |  |  |  |
| --- | --- | --- | --- |
| Обучающийся | \_\_\_\_\_\_\_\_\_\_\_\_\_ | \_\_\_*\_\_\_\_\_\_\_\_\_Торкин Данила Андреевич*\_\_*\_\_\_\_\_\_\_\_\_\_* | |
|  | *Подпись* | *Фамилия Имя Отчество* | |
| Шифр |  | 21К0563 |  |
| Группа |  | КМБО-02-21 |  |
|  |  |  |  |
| Руководитель  работы | \_\_\_\_\_\_\_\_\_\_\_\_\_ | \_\_\_*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*\_\_\_*\_\_\_\_\_\_\_*\_\_\_*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_* | |
|  | *Подпись* | *Фамилия Имя Отчество* | |

Москва 2022

Содержание

[Введение 3](#_Toc121853785)

[Постановка задачи 4](#_Toc121853786)

[Архитектура программы 5](#_Toc121853787)

[Сборка программы 9](#_Toc121853788)

[Руководство пользователя 10](#_Toc121853789)

[Заключение 12](#_Toc121853790)

[Список использованной литературы 13](#_Toc121853791)

# Введение

В рамках курсовой работы необходимо сделать копию игры "Змейка", попутно повышая навык написания кода на C++. Наиболее известна её версия от Nokia, выпущенная в 1997 году. Игрок управляет змеёй, которая ползает по ограниченной стенками плоскости собирая еду. При столкновении с собственным хвостом или краями игрового поля игра завершается. Каждый раз, когда змея съедает кусок пищи, она становится длиннее, что постепенно усложняет игру.

Для осуществления проекта, необходимо было проанализировать механику игры и выбрать инструменты для её создания.

В данном отчете описан процесс создания игры, руководство пользователя и документация кода игры.

# Постановка задачи

Основная задача игрока в этой игре – управлять змейкой на игровом поле, собирая бонусы «яблоки». Змейке нельзя врезаться в свой «хвост». В режиме игры со стенами также нужно избегать стен.

Игровое поле – 475 квадратов размером 32\*32 пикселя. Ячейка может быть яблоком, травой, змейкой или стеной.

Игрок – наблюдает за происходящим сверху, управляя действиями змеи.

Препятствия – хвост змеи, постепенно увеличивающийся в размере, а также стены, если выбран соответствующий режим игры.

Механика игры состоит в том, что змея может передвигаться влево, вправо, вверх, вниз по ячейкам с травой, при этом она не может остановится. Развернуться не сделав крюк змея также не может (то есть нельзя во время того, как ты едешь вперёд резко поехать назад). Собрав яблоко змея увеличивается на один квадрат, а на игровом поле в случайном месте появляется ещё одно яблоко. Если врезаться головой змеи в квадрат хвоста змеи или в квадрат стены, то игра закончится. При игре без стен, если змея достигает конца игрового поля, она продолжает движение с противоположной стороны. Цель игры: собрать как можно больше яблок не врезавшись в препятствие. Количество собранных яблок и является результатом игры.

# Архитектура программы

Программа начинает работу из файла main.cpp, в котором создается объект класса игры Game и из файла конфигурации config.ini считываются необходимые пользовательские настройки (разрешение экрана). Игра запускается после выполнения метода run().

Перед описанием класса игры, рассмотрим структуру Context. В ней содержатся: менеджер состояний (StateManager), менеджер ассетов (SceneManager) и окно – объект класса sf::RenderWindow, которое служит для отрисовки графики.

State – базовый класс для всех состояний игры. Он содержит в себе приватный вектор сущностей (объектов класса Entity), а также методы для инициализации состояния, обновления внутриигровых событий, отрисовки происходящего и шаблонные функции для получения вектора сущностей определенного типа (для проверки коллизий или доступа ко всем объектам). Классы, наследующиеся от State, представлены на рисунке 1.

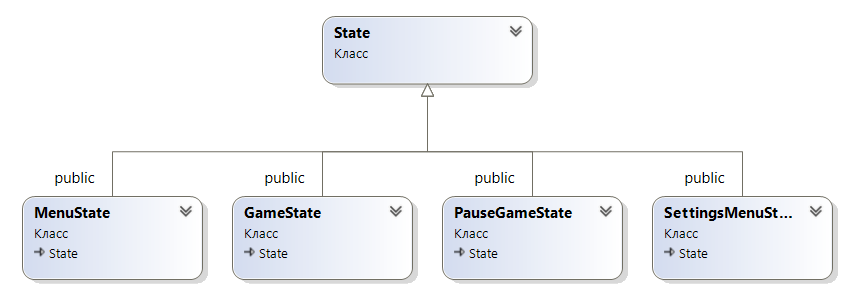


Рисунок 1 – базовый класс State и его производные классы

StateManager – класс, который хранит в себе стек умных указателей unique\_ptr на состояния, т.е. объектов класса State. В нем определены методы добавления, замены и удаления состояний, а также метод получения текущего состояния и метод переключения состояния, который выполняется в начале каждого шага цикла игры. Состояние, которое находится в вершине стека, отрабатывает в текущий момент игры.

SceneManager – класс, который хранит в себе контейнеры std::map с доступными ресурсами, такими как текстуры (sf::Texture) и шрифты (sf::Font). Для каждого вида ресурса определены методы добавления и взятия игрового ассета по ключу-имени.

В классе Game существует приватный умный указатель shared\_ptr на объект Context, который в дальнейшем будет передаваться состояниям для доступа к ресурсам и окна для отрисовки. При запуске игры инициализируется окно и в стек добавляется состояние MenuState, т.е. сначала открывается главное меню. Из него можно попасть в GameState (состояние игры), в SettingsMenuState (меню настроек). При этом состояние меню не удаляется, а остается в стеке.

Класс GameState требует более детального рассмотрения. В его приватной секции находятся данные о номере текущего уровня, данные о самом уровне (вектор строк, считываемый из файла), число доступных на данный момент уровней, информация о ширине и высоте клетки сетки игрового поля, а также булевские переменные проверки выигрыша и проверки текущего состояния ящиков. При инициализации состояния, игровое поле считывается из файла под номером текущего уровня, загружаются необходимые ассеты и создаются игровые объекты с координатами в соответствии с данными об их расположении.

Entity – базовый класс для всех внутриигровых объектов. От него наследуются класс игрока Player, класс цели Goal и класс SolidObject, от которого, в свою очередь, уже наследуются класс стен Wall и класс ящиков Box. Данный класс содержит основные свойства объекта: его спрайт, начальную и текущую позиции, скорость передвижения, параметры анимации (если у объекта имеется как таковая). Диаграмма классов представлена на рисунке 2.

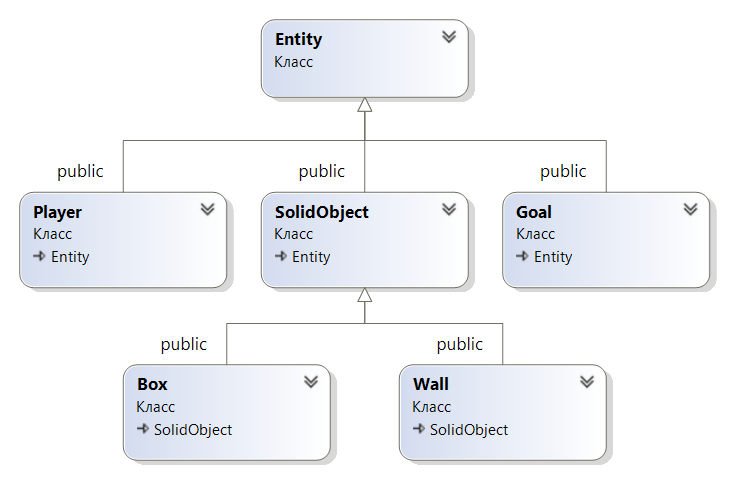


Рисунок 2 – диаграмма классов игровых объектов

У объектов класса Box существуют булевские переменные для проверки, находится ли ящик на цели и доступен ли он в данный момент или нет. Если коробка стоит на обозначенном месте, то его цвет меняется на зеленый, в противном случае – на первоначальную текстуру.

Player – класс персонажа, которым управляет игрок. У игрока содержится основная информация о взаимодействии с объектами класса Wall и Box. Коллизии игрока с другими просчитываются следующим образом: при нажатии кнопки движения, вычисляется текущее положение спрайта, смещенного на несколько пикселей по направлению движения. Далее, получая вектор всех объектов SolidObject, спрайты которых пересекают спрайт персонажа, определяется, можно ли двигаться дальше (если вектор пустой), или же перед нами находится коробка. В случае, если это действительно ящик и на пути он один, то проверяем, может ли он быть смещен на следующую клетку: если да, то «сдвигаем» ящик и увеличиваем счетчик толканий, иначе на пути есть преграда и движение невозможно.

В GameState на каждой итерации цикла игры проверяются состояния коробок и целей. Если хоть один из ящиков находится в таком положении, что сдвинуть его уже невозможно, на экран выводится сообщение об этом и предложение перезапустить уровень. Если же все коробки стоят в обозначенных местах, уровень завершается. На экран выводится сообщение о том, что уровень пройден и через несколько секунд запускается следующий, заменяя текущее состояние GameState на новое, в котором инкрементируется значение текущего уровня. Если уровень последний или игрок сам вышел из игры, GameState удаляется с вершины стека, и игра передает управление MenuState.

# Сборка программы

Перед сборкой программы из исходников, должна быть установлена среда JetBrains CLion, а также подключены последние стабильные версия библиотеки SFML. Инструкции по необходимым настройкам для сборки проекта можно найти на официальных сайтах разработчиков. После завершения сборки, вы можете запустить программу с помощью исполняемого файла Snake.

# Руководство пользователя

При нажатии кнопки «Start» в главном меню, начнется основной игровой процесс. Он продемонстрирован на рисунке 3.

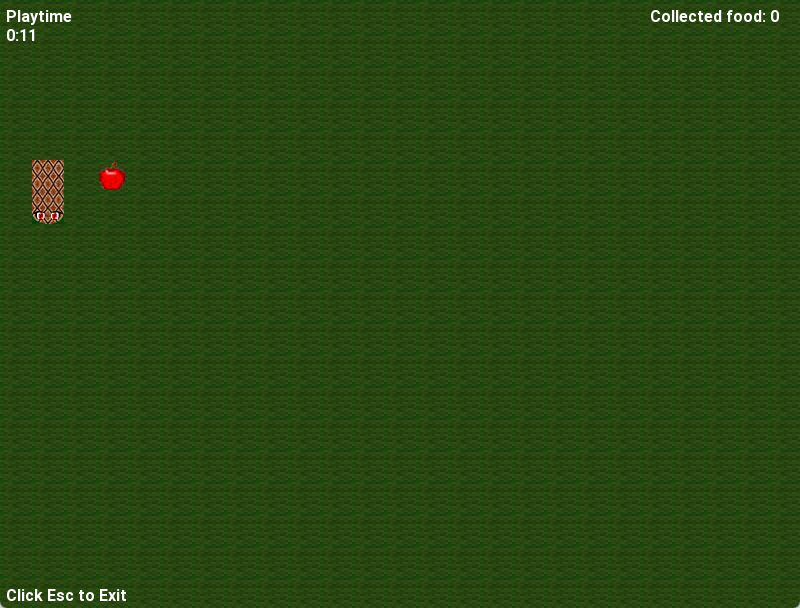


Рисунок 3 – скриншот игрового поля

Перед игроком появится игровое поле, по которому персонаж может двигаться с помощью одной из кнопок движения (стрелочки вверх, вниз, вправо, влево). Чтобы собрать яблоко необходимо наехать на него головой змеи. После этого змея увеличится в размере и игрок услышит звук. На рисунке 4 мы можем увидеть как выглядит змея собравшая 13 яблок в режиме игры со стеной.



Рисунок 4 – скриншот геймплея

В левом верхнем углу мы видим таймер, показывающий сколько идёт игра, а в правом верхнем углу показывается текущий счёт.

При нажатии кнопки Esc игрок может вернуться в меню (длина змеи становится исходной, а счётчик собранных яблок обнуляется).

При столкновении змеи со стеной или со своим хвостом игрок видит экран проигрыша, откуда можно перейти в главное меню или выйти из игры. При проигрыше мы также слышим характерный звук. Также в этот момент игра сохранят результат и соотносит его с таблицей рекордов.

Из главного меню по нажатию кнопки «Leaders» мы попадаем на таблицу рекордов, сформированную из лучших результатов по количеству собранных яблок. Чтобы выйти из таблицы рекордов необходимо нажать «Esc». При перезагрузке игры результаты также сохраняются.

Также в главном меню можно выбрать один из двух режимов игры: «Classic» без стен, «Walls» со стенами.

По нажатию клавиши Escape в главном меню или закрытию окна, игра завершает свою работу.

# Заключение

В данной работе мною удалось выполнить основную задачу курсового проекта – создать клон игры «Змейка» с дополнительными механиками. Во время разработки, я в определенной степени изучил библиотеку SFML, получил опыт в сфере разработки игр и закрепил знания, полученные в курсе «Методы и стандарты программирования».

# Список литературы

Raimondas Pupius SFML Game Development By Example. - Packt Publishing, 2015. - 522 с.

Майкл Доусон Изучаем С++ через программирование игр. - Питер СПб, 2021. - 352 с.

Documentation of SFML 2.5.1:

<https://www.sfml-dev.org/documentation/2.5.1/>