Белорусский государственный технологический университет

Факультет информационных технологий

Кафедра программной инженерии

Проект «Змейка на время»

По дисциплине «Технологии разработки программного обеспечения»

На тему «Технологии разработки ПО. Командная работа над проектом »

Выполнили:

Студенты группы 1

Очной формы обучения

Факультета ИСиТ

Ответственный за проект:

Лопатнюк П.В.

Участники:

Добрего Ю.К.,

Аксеневич И. Р.,

Водчиц А.В.,

Михнюк Н.В.,

Немкович А. В.

Научный руководитель:

ст. преп. Наркевич А.С

2022, Минск

Содержание

1 Введение. Цель и область действия

Цель: командное выполнение проектирования и разработки игры “Змейка” с использованием таймера, обозначающего время окончание игры в зависимости от выбранной сложности.

1.1 Общие задачи и область действия

Общие задачи:

1. Освоить принципы командной работы над проектом и ведения полной документации по нему.
2. Осуществить проектирование и разработку проекта в целом и его отдельных модулей.
3. Выполнить распределение ролей и задач между членами команды.
4. Выполнять тестирование и оперативно исправление ошибок.
5. Предоставить готовый продукт.

Технические задачи:

1. Должно быть несколько уровней сложности (чем больше сложность, тем больше скорость змейки и ее длина).
2. Змейка должна собирать яблоки (положение которых на экране генерируется случайно); при этом игра заканчивается при столкновении змейки со своим "хвостом".
3. На каждый уровень сложности выделяется определенное количество времени, которое пользователь должен продержаться, при истечении времени уровень считается пройденным.

Область действия:

Программное средство предназначено для функционирования на рабочих станциях конечных пользователей в качестве приложения.

1.2 Выбор метода реализации задачи

С развитием компьютерной техники кроме «классического» программного обеспечения (ПО) стали появляться и развиваться целый класс игровых программ. Программы подобного плана развивают логическое и стратегическое мышление, а также дает возможность оператору сделать релаксационную паузу. Кроме того, разработка игровых программ является сама по себе достаточно интересным процессом, позволяющим получить дополнительный практикум программирования.

Для удобства совместной работы был создан новый удаленный репозиторий в GitHub.

1.3 Участники проекта (скрам-команда) и их задачи

Имя участника	Роль в проекте	Задачи
Лопатнюк Полина	Скрам-мастер	Координирует процесс создания проекта, распределяет поставленные задачи между участниками, проводит ежедневные собрания. Прописывает модуль проверки указанных правил в игре, отвечает за общее описание процесса разработки.
Водчиц Анастасия	Член скрам-команды	Отвечает за определения целей и области действия проекта, прописывает модуль прорисовки и генерации бонусов (“яблок”) в игре.

Немкович Анастасия	Член скрам-команды	Отвечает за постановку вариантов использования проекта, прописывает модуль прорисовки консоли и ее границ в игре.
Добрего Юлия	Член скрам-команды	Отвечает за определение используемой технологии и требований к данной системе, прописывает модуль прорисовки змейки и поля игры, а также модуль очистки поля.
Михнюк Никита	Член скрам-команды	Проводит тестирования, отвечает за внедрение таймера в программу и сборку всех модулей в единое целое.
Аксеневич Ирина	Член скрам-команды	Отвечает за подведение итогов выполненного проекта, прописывает модуль движения змейки.

2 Варианты использования

2.1 Варианты использования для бизнес-проектов

Приложение “Змейка”, может быть размещено на различных игровых платформах (таких как: App Store, Google play и др.) Для реализации данной задачи нужно будет действовать по следующей инструкции:

1. Создать аккаунт от имени которого будет выложено наше приложение
2. Разработать и подготовить должным образом политику конфиденциальности и условия пользования
3. Подготовить маркетинговые материалы
   a) Иконка приложения
   b) Скриншоты из приложения
   c) Android application package (формат файла для загрузки приложения)
   d) Баннер

e) Текст-описание приложения

f) Промо Ролик

4.Обеспечить сборку наличием сертификата цифровой подписи

5.Настроить оплату за пользование приложения. (three hundred bucks)

6.Отправить приложение на кроссплатформенные площадки

Для выполнение данных целей требуется материальный капитал, который можно получить при заключении рекламного сотрудничества либо же заключении франшизы. Таким образом мы сможем открыть наше приложение для новых инвесторов, а впоследствии и большого количества пользователей нашей игры.

2.2 Диаграмма использования

3 Используемая технология

3.1 Требования к данной системе

3.1.1 Требования пользователя к программному изделию

Пользовательский интерфейс - это средства общения пользователя с вашей программой. которые могут включать в себя изображения, звуки и текст. Ориентируясь на среднего пользователя интерфейс должен быть простым и удобным. Это снизит вероятность ошибок.

3.1.2 Входные и выходные данные

Входные данные в программе отсутствуют.

Выходными данными должна являться информация о количестве набранных очков и прохождении временного лимита уровня либо о поражении.

3.1.3 Программные ограничения, совместимость

Программное изделие должно работать в операционных системах семейства Windows. Для переноса программы не должны требоваться специальные программные или аппаратные средства.

3.1.4 Результирующие компоненты изделия

В программное изделие должны входить следующие компоненты:

- Командный файл для запуска приложения;
- Программная документация на изделие.

3.1.5 Носители информации

Программа находится на диске.

3.1.6 Требования к надежности

Программный продукт должен функционировать на всех разработанных тестах. Тесты требуется разработать на этапе рабочего проекта.

3.1.7 Требования к составу и параметрам технических средств

Для работы программного модуля необходимо:

- 132 Кб свободного места на жестком диске,
- 1 Мб свободной оперативной памяти.

4. Другие требования

4.1 Описание модулей

Проект состоит из множества модулей, каждый из которых имеет свою собственную функциональность, описанную ниже.

- **Cpp файл Project_snake.cpp**

Project_snake - main(основной) файл проекта. Данный файл объединяет в себе все модули программы и является основным во всей модульной сетке.
В этом файле задаются основные параметры: размер игровой сетки, массив символов для отрисовки, скорость игры, начальные координаты яблок, вектор пар, который содержит координату каждой части змейки. В начале файла идет присваивание начальных данных змейки и добавление 3 частей тела(координат) змейки. После идет создание пред игрового меню, в котором пользователь может выбрать уровень. Каждый уровень отличается своей длительностью по времени и из-за этого сложность пройти данный уровень увеличивается. Также сложность игры увеличивается с каждым съеденным яблоком из-за чего скорость змейки увеличивается.
После задания времени для каждого уровня создается таймер и инициализируется по 0, для того чтобы в следующей итерации цикла таймер начинался заново.

После всех прописанных дополнений для игры идет написание(запуск) самой игры. Это происходит при помощи цикла while. В данном цикле прописаны все возможные происходящие события для змейки, например: если змейка врезалась сама в себя то установить в переменную check значение false, также в это цикле подключатся все другие модули программы, создается задержка для змейки, чтобы она не двигалась слишком быстро, еще в этом цикле находится тело нашего таймера(засекание времени и проверяет его значение).
После этого цикла идет вывод окна где прописывается победил ты или нет, если победа - то выводится Win и указывается имя победителя, если проиграл - выводится GAME OVER.

- **Заголовочный файл AppleGeneration.h**

Заголовочный файл содержит в себе переменные, отвечающие за размер сетки и начальные координаты яблока. Данные координаты образуются за счет функции newapple(), которая прописана в подключенном cpp файле AppleGeneration.cpp.

- **Cpp файл AppleGeneration.cpp**

Файл отвечает за рандомную генерацию координат появления яблок. Координаты яблока по оси X - переменная applex, координата по оси Y - переменная appley. Координаты яблока должны определяться случайным образом в области поля игры. Для этого будем определять их как остаток деления на ширину/высоту поля. (Чтобы избежать генерации яблок на границе необходимо их вычесть из уравнения).

- **Заголовочный файл DrawField.h**

Заголовочный файл объединяет файлы для выполнения основных функций: прорисовки границ, прорисовки змейки, прорисовки яблок, очищения поля. Переменные applex и appley, объявленные в данном файле, используются для рандомной генерации в AppleGeneration.cpp. Прорисовка яблок будет происходить

за счет функции drawApple(), которая прописана в подключенном cpp файле DrawApple.cpp.
Массив field[19][19], объявленный в данном файле позволяет нам выполнить корректную прорисовку границ данного поля drawBorder(), а также его очистку cleanField() . Данные функции прописаны в файлах с расширением cpp : DrawBorder.cpp и CleanField.cpp

- **Cpp файл DrawApple.cpp**

Для прорисовки яблок используется массив для отрисовки field[19][19], содержащийся в заголовочном файле DrawField.h. В качестве индексов элементов используются полученные из AppleGeneration.cpp координаты генерации яблок. Данному элементу присваивается значение char(178), что означает ▓ символ (это и есть наше яблоко).

- **Cpp файл DrawBorder.cpp**

Файл отвечает за рисование границ. Благодаря функциям прописанным в данном файле, клетки выходящие на одну клетку за пределы field[19][19], заполняются символом ▓ , в качестве границы.

- **Cpp файл DrawInConsole.cpp**

Файл отвечает за оформление консоли. Изначально мы устанавливаем начальные точки наших координат. Функция при прохождении через все точки на нашем поле, проверяет есть ли на данных координатах, змея или яблоко.

1. Если есть яблоко - цвет красный
2. Если на этих координатах находится змея - цвет зеленый
3. Если ничего - цвет черный

Также в консоли выводится
счет игры snake.size() - 3
скорость змейки 80 / speed

- **Cpp файл CleanField.cpp**

Функция выполняет очищения поля. Все клетки заполняются пробелами - то есть делает их пустыми.

- **Cpp файл DrawSnake.cpp**

В данном файле прописан код для прорисовки тела змейки. Вектор snake хранит в себе пару чисел - координаты n-ной части тела змейки. Клетке поля, соответсвующей координатом положения части тела змейки присваивается значение char(177). Прорисовка происходит с учетом текущей длины змейки, причем голова оформлена другим символом -char(178).

- **Заголовочный файл SnakeMovement.h**

Заголовочный файл содержит в себе переменные, отвечающие за размер сетки, начальные координаты яблока, скорость игры и вектор пар, который содержит координаты каждой части змейки. Данные координаты образуются за счет функции moveSnake(), которая прописана в подключенном cpp файле SnakeMovement.cpp.

- **Cpp файл SnakeMovement.cpp**

Данный модуль отвечает за передвижение змейки. В зависимости от нажатой клавиши направления мы изменяем координаты точек нашей змейки, а если клавиша не была нажата, то просто смещаем координаты нашей змейки на 1 клетку поля в ту сторону, в которую движется змейка. Если голова нашей змейки съела яблоко, то увеличиваем нашу змейку, т.е. добавляем еще одну координату и создаём новое яблоко, также меняем скорость нашей змейки.

- **Cpp файл CheckRules.cpp**

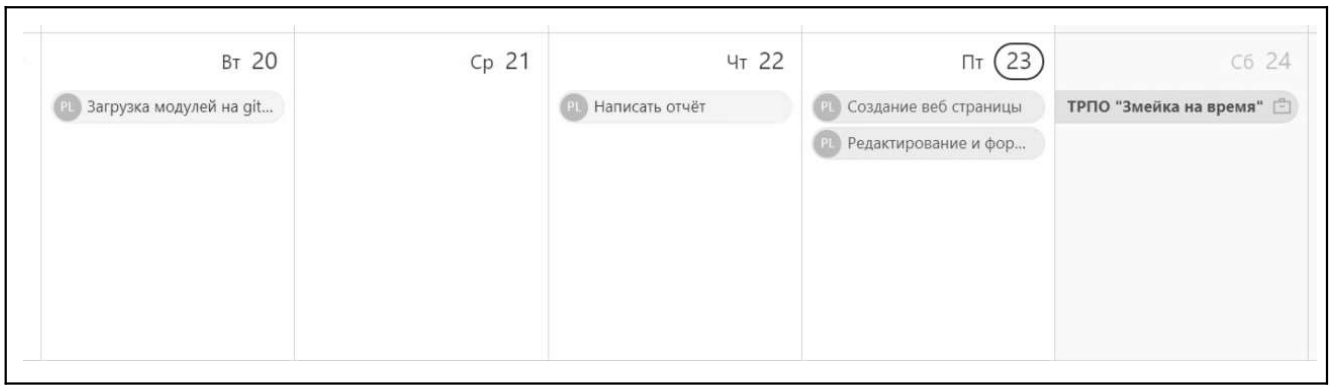
Модуль проверяет, не врезалась ли змейка сама в себя, а тем самым контролирует соблюдение главного правила игры. Данный процесс реализуется при помощи сверки положения частей змейки на поле. При совпадении каких-либо координат положения частей змейки, возвращается значение false, что останавливает цикл, реализующий игру.

- **Заголовочный файл CheckRules.h**

Данный файл включает в себя вектор пар, который содержит координату каждой части змейки. А также содержится булевая функция, находящаяся в CheckRules.cpp, что отвечает за ход игры.

4.2 Описание процесса разработки

Во время работы над проектом была использован наиболее распространенная методология разработки программного обеспечения - Scrum, которая является формой методологии гибкой разработки.
Была создана scrum-команда из 6 человек, в которой были определены роли Scrum-роли каждого участника. После была началась работа над реализацией проекта в рамках одного sprint.
Scrum-мастер провёл начальное ознакомление с проектом, который предстоит реализовать, и определил назначение, цель проекта, модули, входящие в него.
После первого ознакомления в проектом была создана scrum-доска для обозначения главных целей команды. Более подробно задачи каждого были обсуждены во время scrum-встречи.

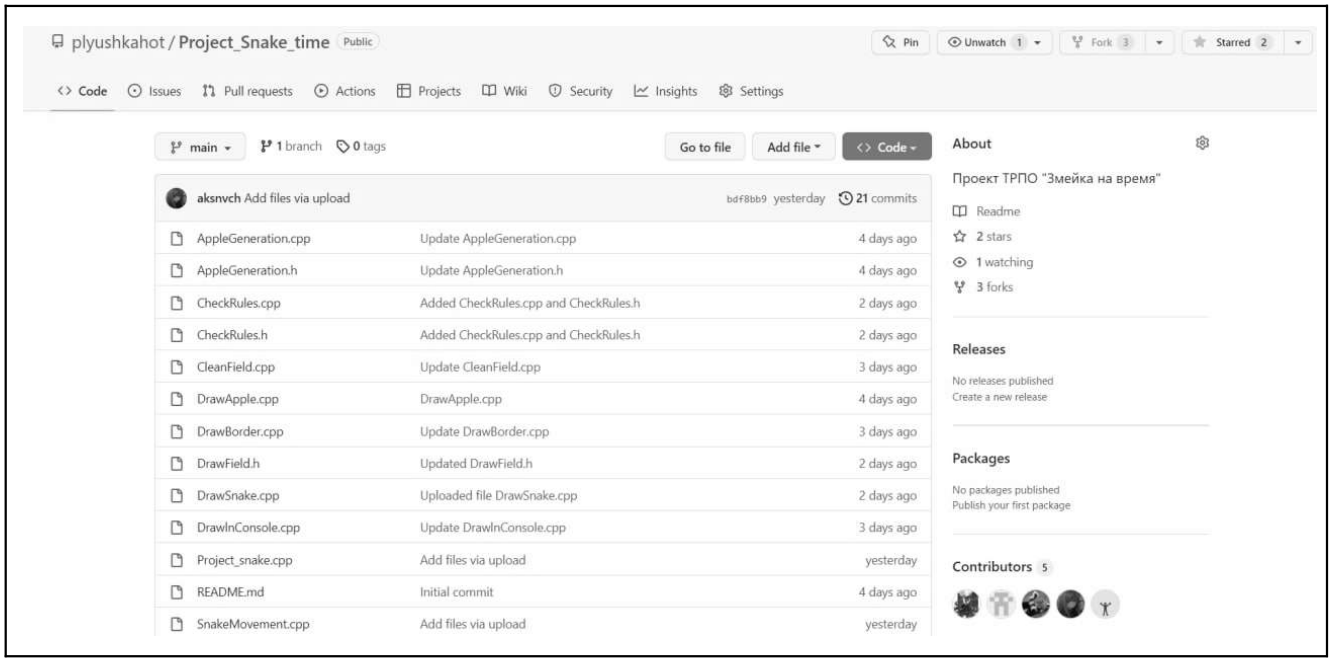


Также во время scrum-встреч было создано краткое описание основной концепции разрабатываемого проекта.

Создание проекта.

Для удобства работы над проектом был создан новый удаленный репозиторий в GitHub. Каждый член команды был добавлен и получил права доступа.

Были распределены модули среди участников проекта. А также был составлен список пользовательских историй, описывающих необходимую функциональность каждого модуля.



5 Тестирование

В ходе тестирования игры, каждый уровень был пройден не один десяток раз. В тестировании проверялись возможность ошибок в размещении объектов игры, таких как яблоки. Яблоки всегда оказывались на доступных местах для змейки. При тестировании динамики передвижения змеи также не было замечено никаких ошибок, змея все время поворачивала в ту сторону в какую была нажата клавиша. При поглощении яблока размер змейки всегда увеличивался, а при столкновении ее самой с собой следовало GAME OVER. Скорость передвижения всегда увеличивалась по определенному заданному значению.

Также проводилось тестирование на разных компьютерах с разными версиями ОС Windows. Везде игра запускалась, шла с одинаковой скоростью.

При запуске нескольких экземпляров игры не замечено багов, однако играть в таком случае не представляется возможным. При этом не было никаких тормозов.

Одной из основных проблем, где наша команда столкнулась с багами, является внедрение таймера. При окончании времени игры одного уровня и перехода на новый уровень размер змейки не обнулялся, а оставался прежним. Также стоит упомянуть, что слетали настройки прорисовки змейки и границ при внедрении таймера, а вместо них был знак “?”. Иногда не запускалась сама игра.

Но все разносторонние проблемы быстро исправлялись человеком отвечающим за определенный файл.

В тестировании были задействованы не только люди строго работающие над проектом. Все недочеты, которые были выявлены не основной командой также быстро исправлялись.

6. Выводы

Для создания проекта была разработана программа, реализующая классическую игру “Змейка” на языке C++.

Уровень в игре считается пройденным, если змейка не врежется сама в себя на протяжении установленного времени, конец игры предусмотрен в случае гибели змейки.

В процессе написания программы были рассмотрены все варианты неправильной работы программы, к примеру: не находится ли голова змейки на яблоке или не пресекает ли голова змейки заданное игровое поле. Все эти проблемы были выявлены и благополучно устранены.

Наша команда упорно трудилась над созданием этого проекта и устранением ошибок.