Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

**Курсовой проект по дисциплине**

**«Операционные системы»**

**Интерактивная клиент-серверная игра**

Студент: Пермяков Никита Александрович

Группа: М8О –208Б-19

Вариант: 4

Преподаватель: Миронов Е.С.

Оценка: \_\_\_\_\_\_\_\_\_\_\_

Дата: \_\_\_\_\_\_\_\_\_\_\_

Подпись: \_\_\_\_\_\_\_\_\_\_\_

Москва, 2020

**Содержание**

1. Постановка задачи
2. Описание программы
3. Сведения о программе
4. Метод и алгоритм решения
5. Описание игровой логики
6. Демонстрация работы программы
7. Технологические требования
8. Вывод

**Постановка задачи**

Реализовать интерактивную многопользовательскую игру с архитектурой «Клиент – Сервер». В данной системе должно существовать 2 вида узлов: «управляющий» и «вычислительный».

Наименование игры:

NodeNetwork

**Описание программы**

1. Начальное меню при запуске программы

- выбор роли: клиент либо сервер.

- режим окна: полноэкранный, либо оконный (изменение по нажатию)

- выход

2. Меню сервера

- установка количества игроков

- кнопка запуска игровой сессии

- отображение количества игроков

3. Комната ожидания игроков

- строка ввода хоста (автодополнение – пушка)

- список доступных серверов

4. Игровое поле

**Сведения о программе**

Программа состоит из восьми основных файлов и библиотеки, реализующей взаимодействие с текстурами - SFML.

1) main.cpp – запуск основного «игрового» цикла. Установка генератора случайных чисел на основе времени. Установка параметров стартового окна.

2) MainMenu.сpp – файл описания интерфейса главного меню.

3) Game.cpp – обработка действий пользователя в главном игровом цикле.

4) ClientMenuServerList.cpp – интерфейс клиента во время подключения к серверу. Обработка событий при вводе port пользователем.

5) Client.cpp – обработка событий со стороны клиента, установка соединения. Подключение к игровому полю.

6) Cell.cpp – файл, описывающий юнит – увеличение в зависимости от времени и переменной clock (класс «значимости», измеряющий полезное время. Свойсто моннотонности функции).

7) Gameboard.cpp – установка параметров игрового поля. Случайная установка параметров у юнитов: местоположение, емкость, класс и размер. Отрисовка юнитов и связей.

8) Link.cpp – отрисовка стрелок, «перетекание» характеристики от юнита к юниту, с помощью взаимного измениения двух противоположных полей у каждого. Рокировка связи по требованию обработчика событий. Конечный автомат перехода состояний.

9) Manager.cpp – файл линковки текстур. Для одноразовой загрузки, применение по указателю на функцию.

10) Player.cpp – описание начальных юнитов, доступных для игроков.

11) Server.cpp – устанвка UD протокола для ожидающих клиентов. Смена на однопользовательский режим в зависимости от установки клиента в комнате ожидания. Перебор сокетов и обработка статуса ответа. Установка соединения связей между юнитами.

12) sfml-\*-.dll – файлы библиотеки динамической компоновки.

13) SFML\\* - файлы фреймворка, интерфейсы для структур, обработки событий, звука и тд.

14) sfml-\*-.lib – файлы библиотеки, статические.

15) textures\\* - текстуры

**Метод и алгоритм решения**

1. Определяется архитектура приложения.

Модель распределенного представления данных (модель сервер терминалов) не подходит по условию задачи. Пример такой игры – одно вычислительное устройство (мейнфрейм) и несколько устройств ввода. В игре может быть реазизована обработка одной клавиатуры с разными участками клавиш для каждого игрока.

Файловый сервер также не подходит, по причине асинхронной, либо последовательной обработки запросов от пользователей, в то время, когда нужно синхронно, даже если будет репликация состояний.

Реализация прикладного компонента на стороне сервера, то есть сервер приложений – наиболее подходящая архитектура для данной задачи. Перенос функций прикладного компонента на сервер снижает требования к конфигурации клиентов и упрощает поддержку, но представляет больше требований к производительности сервера.

Трехзвенная архитектура, когда участвует более одного сервера – не подходит из-за сложной реализации.

1. Определяется тип клиента

В нашем случае – «тонкий» клиент. Когда состояние, хоть и хранится на стороне клиента, но может быть посчитано на сервере. Тем самым, клиент воспроизводим сервером.

3 Проектируются сущности – классы, и описываются интерфейсы.

**Описание игровой логики**

Игровой цикл считается от момента запуска со стороны сервера, до захвата всех юнитов на игровом поле одним из игроков.

Каждый юнит имеет случайные характеристики:

* цвет - для каждого игрока разный, но одинаковый для его юнитов. Не изменяется в процессе игры. В начальном состоянии – серый.
* Числовую характеристику - сила юнита. Для каждого юнита случайна в начальном состоянии.
* Емкость – максимально возможная сила юнита, для каждого юнита случайна и не изменяется в процессе игры.

Начальное распределение юитов среди игроков случайно по цвету, размеру, силе, емкости и местоположению, но одинаково по количеству. Сохранен игровой баланс.

(Побеждает чаще тот, кто забирает как можно раньше максмальное количество любый незанятых юнитов. То есть выбирать для захвата в первую очередь необходимо самые слабые, в порядке возростания силы. Не тратить на изменение уже занятого юнита. Емкость влияет на ситуацию только локально по времени и расположению).

Может возникнуть равносилная ситуация, при которой количественные показатели каждого юнита одинаковы для двух и более игроков. В этом случае «претекание» характеристики разных цветов невозможно. Тогда объявляется ничья. (Программа не обрабатывает этот случай – предполагаемое решение – ввод внутриигрового времени, ограничивающий сессию сверху). Все юниты, в конечном состоянии, будут закрашены одним цветом, который определяет победителя.

Переход силы между юнитами осуществляется по направлению стрелки, увеличивается у того, на кого она указывает, уменьшаясь на столько же единиц с такой же скоростью и ускорением у противоположного юнита.

Измениение цвета наступает при состоянии с характеристикой силы равной 0. Дальнейшее изменение характеристики идет на увеличение. Она не может быть отрицательной.

Установка стрелки – связи, между юнитами по левому щелчку мыши, путем drop drap от указывающего до указанного юнита. Возможно снятие стрелки – связи, между юнитами, по левому шелчку мыши и изменение направления на противоположное по правому. Тачпад не поддерживает.

Учасники равноправны между собой, сервер – один из учасников.

**Технологические требования:**

Тестировалось:

- Windows 10 Pro

- Core i7 10th gen (1999 Mhz)

- 4 Gb GUM NVideo 230mx

- size screen: 1698 x 1968 px (400 ppi)

- 16 Gb RAM

**Демонстрация работы программы**

Репозиторий: <https://github.com/nikit34/NodeNetwork_game_sfml>

Видео-отчёт: <https://youtu.be/CHHReqf70VA>

**Выводы**

Основная идея архитектуры «клиент-сервер» - разделение сетевого приложения на несколько компонентов, каждый из которых реализует свою логику. Компоненты такого приложения могут выполняться на разных компьютерах, выполняя функции сервера или клиента.

**Преимущества**

* Нет дублирования кода программами сервера и программами клиентами.
* Так как все вычисления выполняются на сервере, то требования к компьютерам, на которых установлен клиент, снижаются.
* Повышенная защищенность сервера
* Контроль полномочий, доступ к данным только клиентам с соответствующими правами доступа.

**Недостатки**

* Неработоспособность сервера может сделать неработоспособной всю вычислительную сеть, (не хватает вычислительных мощностей на всех клиентов).
* Высокая стоимость устройст для сервера.