# 

ЗМІСТ

[ВCТУП 6](#_Toc500980553)

[1 ПОСТАНОВКА ПРОБЛЕМИ 8](#_Toc500980554)

[1.1 Постановка технічного завдання 8](#_Toc500980555)

[1.2 Аналіз досліджуваної проблеми 8](#_Toc500980556)

[1.3 Огляд існуючих програмних систем 9](#_Toc500980557)

[2 ПРОЕКТУВАННЯ МОДЕЛІ РОЗРОБКИ 13](#_Toc500980558)

[2.1 Вибір моделі розробки 13](#_Toc500980559)

[2.2 Проектування варіантів використання 14](#_Toc500980560)

[3 РЕАЛІЗАЦІЯ ПРОГРАМИ 18](#_Toc500980561)

[3.1 Вибір мови програмування та засобів для розробки 18](#_Toc500980562)

[3.2 Організація класів 18](#_Toc500980563)

[3.2.1 Клас Mono 19](#_Toc500980564)

[3.2.2 Клас Menu 20](#_Toc500980565)

[3.2.3 Клас Player 20](#_Toc500980566)

[3.2.4 Клас Dice 21](#_Toc500980567)

[3.2.5 Клас Tokens 21](#_Toc500980568)

[3.2.6 Клас Map 22](#_Toc500980569)

[3.2.7 Структура CoordGo 22](#_Toc500980570)

[3.2.8 Структура Coordinate 22](#_Toc500980571)

[3.2.9 Структура DescriptionOfRectangles 22](#_Toc500980572)

[4 ТЕСТУВАННЯ ТА ОЦІНКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 23](#_Toc500980573)

[4.1 Основи тестування програмного забезпечення 23](#_Toc500980574)

[4.2 Особливості тестування ігор 23](#_Toc500980575)

[4.3 Тестування та результат виконання програми 25](#_Toc500980576)

[ВИСНОВКИ 28](#_Toc500980577)

[СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 29](#_Toc500980578)

[ДОДАТКИ 30](#_Toc500980579)

[ДОДАТОК A 32](#_Toc500980580)

[ДОДАТОК B 104](#_Toc500980581)

# ВCТУП

Гра — моделювання іншого виду діяльності із розважальною чи навчальною метою. Гра відрізняється від роботи тим, що не ставить перед собою безпосередньо корисної мети (але користь від неї може мати місце), а також тісно межує з мистецтвом, хоч зазвичай не створює художніх цінностей.

Темою для написання курсової роботи я обрав відому мільйонам людей гру — «Монополія».

За основу гри «Монополія», були взяті правила з подібної гри «The Landlord's Game», запатентованої в 1904 році. Вона була призначена як навчальний інструмент для ілюстрації негативних аспектів концентрації землі в приватних монополіях .

Монополія — популярна настільна гра економічного характеру, опублікована компанією Parker Brothers (підрозділом компанії Hasbro).Чарльз побачив як його сусіди грають гру мета якої заробляти гроші продаючи будинки та готелі. Гравці змагаються за надбання економічних переваг шляхом визначеної правилами гри економічної активності, включаючи купівлю, оренду та торгівлю власністю з використанням «забавкових» грошей.

Відповідно до Hasbro, з моменту як Чарльз Дерроу запатентував гру у 1935 році, близько 750 мільйонів людей зіграли в цю гру, що зробило її найпопулярнішою комерційною настільною грою у світі. Книга рекордів Гінесса від 1999 року має запис щодо попередньої статистики Hasbro про 500 мільйонів людей які грали в Монополію. Журнал «Ігри» посвятив гру Монополію в члени своєї Зали Слави.

Фактично стратегія складається з ігрового поля, яке поділене на квадрати. Вони поділяються на підприємства або активи, події і в’язницю, потрапляючи в яку учасник пропускає хід. Черга ходити визначається кидком кубика – у кого більше, той і перший. Кількість полів за хід, відповідає сумарним очками випали на кубиках. Суть гри монополія полягає в тому, щоб раціонально використовуючи свій капітал на купівлю нерухомості призвести супротивників до абсолютного банкрутства.

Жанр цієї гри — настільна гра. Проте в епоху інноваційних технологій люди надають перевагу онлайн іграм ніж іграм наживо. Для задоволення потреб людини, було розроблено комп’ютерну версію гри монополія, в яку можна грати онлайн з людьми зі всього світу, а також можливо грати з комп’ютером.

# 1 ПОСТАНОВКА ПРОБЛЕМИ

## 1.1 Постановка технічного завдання

Головною метою завдання є реалізувати програмний продукт, який буде слугувати в розважальних цілях для користувача. Для того, щоб реалізувати програму потрібно ознайомитися з логікою гри , а також із моделями життєвого циклу програмного забезпечення.

Визначити цільову аудиторію для застосування продукту, скласти архі-тектуру класів, описати методи класів. Для спрощення написання коду надамо перевагу об’єктно-орієнтованому програмуванню, у зв’язку з тим, що ця пара-дигма програмування використовує об’єктне мислення, яке дозволяє проекту-вати важкі системи. Для цього насамперед потрібно скласти UML діаграму класів, щоб умовно розбити код на певні частини. Після реалізації алгоритмів програми потрібно скласти графічний інтерфейс. З його допомогою взаємодія між користувачем та системою стає набагато продуктивнішою. Адже, як відомо, більшість інформації людина сприймає за допомогою зорових органів.

## 1.2 Аналіз досліджуваної проблеми

Розробка програмного продукту – складний і тривалий процес, тому для початку слід визначити потреби своїх майбутніх користувачів, функції, які додаток повинен виконувати, а також, визначитися з його типом і операційною системою.

Рoзрoбляємo інтерфейс програми для гри «Монополія» засобами мови програмування C++ з допомогою графічної бібліотеки SFML.

Перед тим, як почнеться розробка програмного продукту, слід чітко визначитися що саме вам потрібно. Від того, які функції має додаток, залежить тривалість його розробки, а також, рівень складності.

Програма виконуватиме наступні фунції:

- керування за допомогою курсору та правої кнопки миші;

- вибір кількості гравців і їх фігурки;

- підкидання гральних кубиків;

- пересування фігурки гравця по карті;

- купівля підприємства;

- можливість обміну підприємств з доплатою або без;

- зняття певної суми з рахунку гравця, який зупинився на підприємстві іншого гравця;

- отримання картки «Шанс», або «Біржа» при зупинці в полі їх видачі;

- арешт, при зупинці в полі «Go to Jail»;

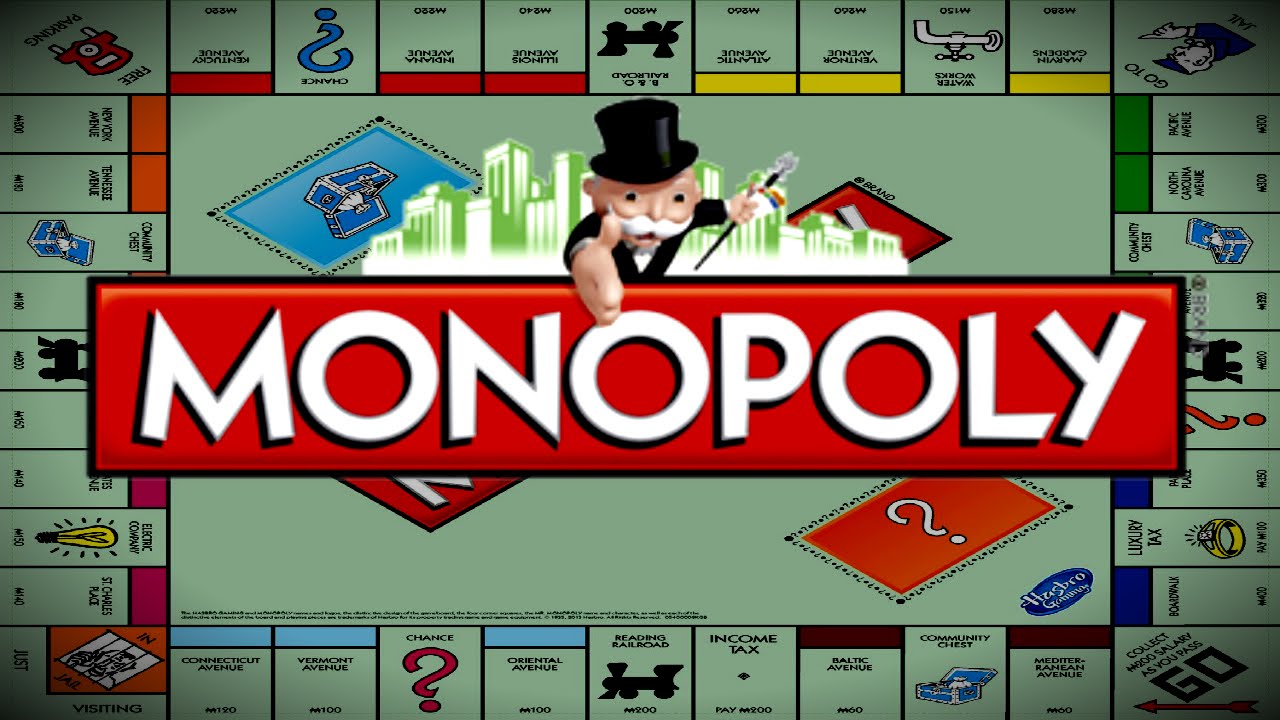
- вихід з в’язниці при випаданні на двох кубиках однакового числа;

- виведення вікна з текстом у разі поразки чи виграшу.

## 1.3 Огляд існуючих програмних систем

Правила гри : учасники по черзі кидають пару кубиків і роблять відповідну кількість ходів на полі. Зайнявши клітку з нерухомістю, гравець може купити її за умови, що актив вільний, або заплатити податок за прейскурантом за відвідування даного поля того, кому належить підприємство. Якщо учасник зайняв клітку з подією, він отримує особливу вказівку.

Гра «Монополія» реалізована, практично, на всіх відомих операційних системах, зокрема: Windows, Mac Os, IOS, Android.

Рисунок 1.1 версія гри на OS Android

Дивлячись на те, що Монополію грають у всьому світі, карти цієї гри можуть момент існує багато варіацій карт. Цікавіше грати, якщо підприємства на карті безпосередньо розташовані в твоїй країні і ти їх знаєш.

Дана гра є дуже захоплюючою та цікавою. Зазвичай гра триває декілька годин. Все залежить від суперника, і обраної ним стратегії . Проте, мета цієї курсової роботи: підсумувати знання з мови програмування C++, навчитися застосувати об’єктно-орієнтований метод програмування, а також опанувати створення графічного інтерфейсу. Тому була вибрана саме ця тема для написання роботи, щоб показати наглядно навики, зібрані за 3 семестри навчання.



Рисунок 1.2 – Онлайн версія гри

# 2 ПРОЕКТУВАННЯ МОДЕЛІ РОЗРОБКИ

## 2.1 Вибір моделі розробки

Модель життєвого циклу - це структура, що складається із процесів, робіт та задач, які включають в себе розробку, експлуатацію і супровід програмного продукту; охоплює життя системи від визначення вимог до неї до припинення її використання. На сьогодні найбільшого розповсюдження набули дві моделі:

- каскадна;

- спіральна.

В каскадній моделі не можна внести зміни до попередніх процесів розробки, але відома кількість ітерацій, тобто етапів розробки. Тому для невеликих проектів ця модель повністю підходить.



Рисунок 2.1 Каскадна модель розробки

Спіральна модель. Кожен виток спіралі відповідає створенню фрагмента або версії програмного забезпечення, на ньому уточнюються цілі і характеристики проекту, визначається його якість і плануються роботи наступного витка спіралі. Таким чином поглиблюються і послідовно конкретизуються деталі проекту і в результаті вибирається обґрунтований варіант, який доводиться до реалізації.

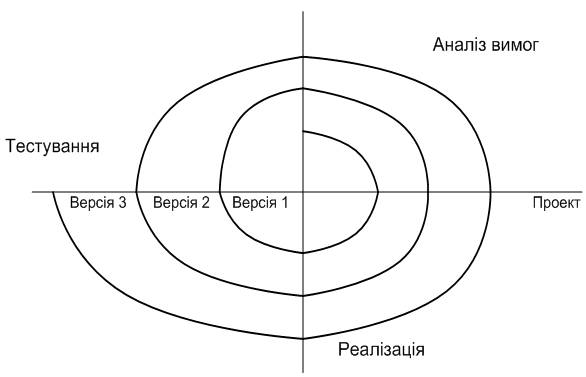


Рисунок 2.2 - Спіральна модель розробки

Після того як вимоги повністю визначені, відбувається перехід до проектування, в ході якого створюються документи, що детально описуються для програмістів спосіб і план реалізації зазначених вимог. Після того як проектування повністю вионане, програмістами виконується реалізіція отриманого проекту.

На наступній стадії процесу відбувається інтерграція окремих компонентів, що розробляються різними командами програмістів. Після того як реалізація і інтеграція завершені, проводиться тестування і налагодження продукту; на цій стаді усуваються всі недоліки, що з’явилися на попередніх стадіях розробки. Після цього програмний продукт впроваджується і забезпечується його підтримка – внесення нової функціональності і усунення помилок.

## 2.2 Проектування варіантів використання

Визначені варіанти використання:

* запустити гру;
* вибрати кількість гравців і їх фігурки;
* кидати кубики;
* купляти підприємства;
* виграти або програти;
* вийти з гри;

Діаграма варіантів використання. Суть даної діаграми полягає в наступному: проектована система представляється у вигляді безлічі сутностей або акторів, які взаємодіють з системою за допомогою, так званих варіантів використання. При цьому актором (actor) або дійовою особою називається будь-яка сутність, що взаємодіє з системою ззовні. Це може бути людина, технічний пристрій, програма або будь-яка інша система, яка може служити джерелом впливу на модельовану систему так, як визначить сам розробник. У свою чергу, варіант використання (use case) служить для опису сервісів, які система надає акторові. Іншими словами, кожен варіант використання визначає певний набір дій, який чинять системою при діалозі з актором. При цьому нічого не говориться про те, яким чином буде реалізовано взаємодія акторів із системою.

У загальному випадку, діаграма варіантів використання є граф спеціального виду, який є графічною нотацією для представлення конкретних варіантів використання, акторів, можливо деяких інтерфейсів, і відносин між цими елементами. При цьому окремі компоненти діаграми можуть бути укладені в прямокутник, який позначає проектовану систему в цілому. Слід зазначити, що відносинами даного графа можуть бути тільки деякі фіксовані типи взаємозв'язків між акторами і варіантами використання, які в сукупності описують сервіси або функціональні вимоги до моделюється системі.

Програма написана для розважальних цілей, тому в першу чергу потрібно дати користувачеві можливість розважитись і відпочити в процесі гри. У зв’язку з цим інтерфейс, а також керування програми мають бути надзвичайно легкими та зрозумілими.

Можна виділити два основних актора – користувачі. Діаграма варіантів використання зображена на рисунку 2.3.

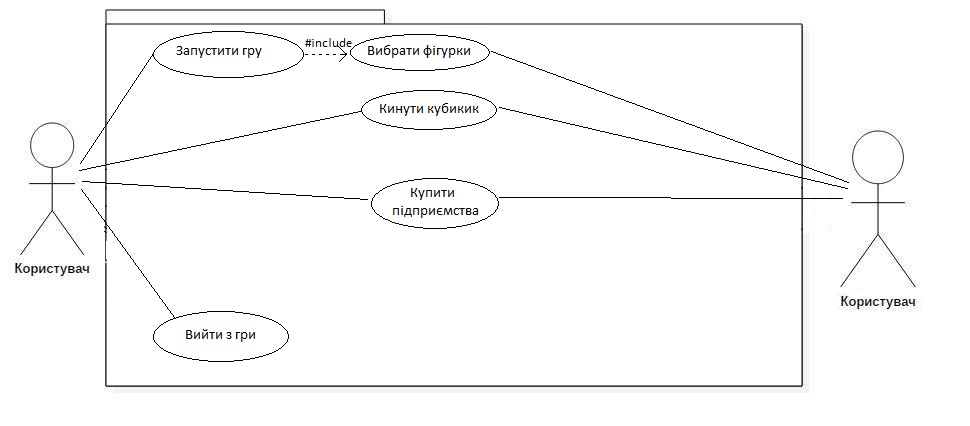


Рисунок 2.3 Діаграма варіантів використання

# 3 РЕАЛІЗАЦІЯ ПРОГРАМИ

## 3.1 Вибір мови програмування та засобів для розробки

Вибір мови програмування – є важливим критерієм при розробці програмного продукту, так як кожна мова програмування має свої особливості, а також різноманітні засоби для вирішення конкретної задачі.

Для написання програми було обрано мову C++. Її суттєвими перевагами є швидкодія. C++ базується на об’єктно-орієнтованому методі програмування, і ця перевага дозволяє працювати із реальними об’єктами.

. Незважаючи та те, що кажуть про застарілість мови C++, на ній і досі продовжують розробляти найрізноманітніші програми для великої кількості платформ і систем, а швидкість роботи програм на С++ не поступається іншим мовам програмування і навіть краща.

Для курсової роботи, мною була вибрана графічна бібліотека SFML (Simple and Fast Multimedia Library).

SFML забезпечує використання 2D-графіки з апаратним прискорення OpenGL. SFML також може бути використаний для OpenGL-проектів. SFML також надає різні модулі для полегшення програмування ігор і мультимедійних додатків.

## 3.2 Організація класів

Провівши аналіз функціоналу програмного продукту, було прийняте рішення для створення наступних класів:

* Mono відповідає за виведення карти і фігур;
* Menu меню гри, а також вікно торгівлі;
* Player клас який зберігає дані гравця;
* Dice підкидання кубиків;
* Tokens дані про фігурки;
* Map містить дані, які взаємодіють з картою;
* Chance відповідають за генерування картки «Шанс»;
* CommunityChest відповідає за генерування куртки «Біржа»

та структур:

* CoordGo взаємодія фігурок з картою;
* Coordinate координати розташування фігурок;
* DescriptionOfRectangles вміщає дані про кожне підприємство;

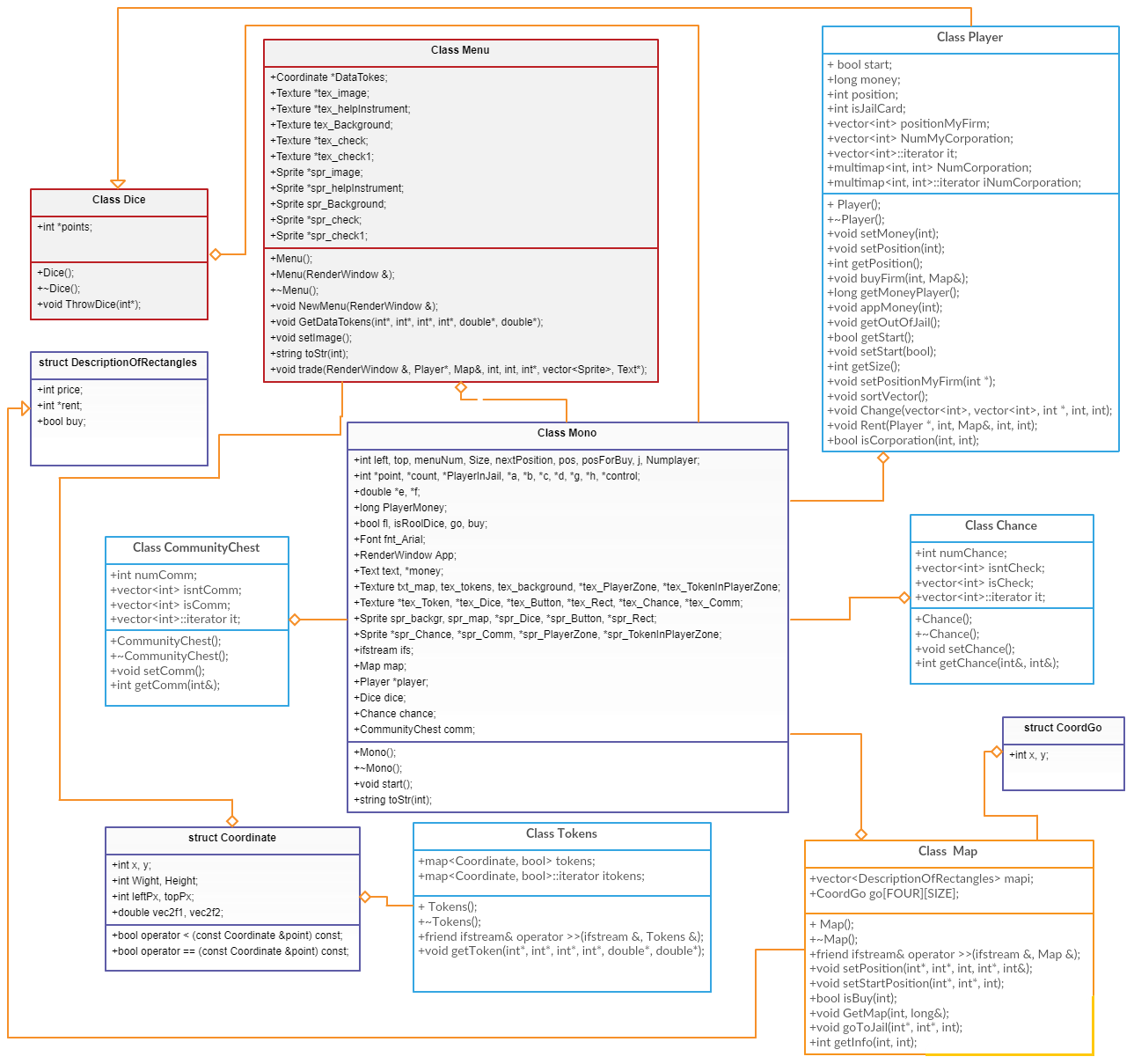


Рисунок 3.1 – UML діаграма класів програми

## 3.2.1 Клас Mono

Опрацьовує все, що виводиться в головному вікні з картою. При покупці підприємства, зафарбовує прямокутник під цим полем, в колір гравця. Переміщає фігурки гравців на певну позицію. Виводить кількість грошей в гравців.

Містить єдиний метод start(), який і починає гру.

## 3.2.2 Клас Menu

Виводить меню, де можна вибрати кількість гравців і їхні фігурки. Кількість гравців повинна два або більше, в противному випадку гра не почнеться. Є 6 фігурок і вони не повторюються. Також в цьому класі є метод, який відповідає за торгівлю. Можна обміняти своє підприємство на підприємство суперника з доплатою чи без неї.

Передбачені наступні методи:

* конструктор;
* конструктор з параметром – вікно, де є вибір почати гру або вийти;
* деструктор;
* NewMenu() – вибір гравців і фігурки;
* GetDataTokens() – отримуємо дані про фігурки;
* setImage() – встановлює зображення для торгівлі;
* trade() – вікно торгівлі.

## 3.2.3 Клас Player

Основний клас користувача. Містить всю інформацію про нього: кількість грошей, на якій позиції знаходиться, на якій позиції знаходяться його підприємства та поле в якому інформація про те чи знаходиться він у в’язниці.

Передбачені наступні методи:

* конструктор;
* деструктор;
* setMoney() – присвоює гроші гравцю;
* setPosition() – встановлює позицію гравця;
* getPosition() – повертає позицію гравця;
* buyFirm() – купівля підприємства;
* getMoneyPlayer() – повертає кількість грошей;
* appMoney() – додає певну суму грошей;
* getOutOfJail() – віднімає гроші при виході з в’язниці;
* getStart() – чи кинув кубики граіець;
* setStart() – гравець кинув кубики;
* getSize() – повертає кількість підприємств, які належать даному гравцю;
* setPositionMyFirm() – додає підприємство, яке купив гравець;
* void sortVector() – сортує підприємства гравця за розташуванням на карті;
* void Change() – при обміні підприємствами з суперником видаляє дане підприємство з списку і додає нове.
* Rent() – стягує гроші в гравця, який зупинився на підприємстві суперника;
* isCorporation() – при покупці всіх підприємств одного кольору, вони стають корпораціями і рентна плата збільшується;

## 3.2.4 Клас Dice

Клас, який генерує кількість очок, які викинуть кубики. Крім конструктора та деструктора містить один метод ThrowDice(), який і генерує кількість очок.

## 3.2.5 Клас Tokens

Пов’язаний з фігурками. Містить поля, в яких зберігається координати прямокутника з фігуркою.

Передбачені наступні методи:

* конструктор;
* деструктор;
* перегружена операція читання з файлу, яка записує в поле tokens координати фігурки;
* getToken() – функція, яка забезпечує вибір фігурки в меню.

## 3.2.6 Клас Map

Містить всі дані про карту гри, про ціни підприємств, рентну плату та все інше. Також містить координати, по яких переміщаються фігурки, щоб стояти в центрі підприємств.

Передбачені наступні методи:

* конструктор;
* деструктор
* перегружена операція читання з файлу, яка записує в масив координати ходів та відомості про підприємства;
* setPosition() – метод який дозволяє фігурці переміщатися покроково, по кожному полю, а не зразу переходити на поле, яке випало на кубику;
* setStartPosition() – встановлює початкову позицію для фігурок;
* isBuy() – чи поле вже є власністю якогось гравця;
* void GetMap() – купівля підприємства;
* void goToJail(int\*, int\*, int) – при зупинці в полі «Go to jail», відправляє граця у в‘язницю.
* getInfo() – при потраплянні на підприємство суперника ця функція знімає гроші в одного гравця і віддає їх власнику підприємства;

## 3.2.7 Клас Chance

Генерує карту «Шанс»

## 3.2.8 Клас CommunityChest

Генерує карту «Біржа»

## 3.2.9 Структура CoordGo

Зберігає координати, по яким переміщаються фігурки на карті.

## 3.2.10 Структура Coordinate

Зберігає координати прямокутника фігурки. Містьти перевантажений метод «<» та «==» для сортування в контейнері map.

## 3.2.11 Структура DescriptionOfRectangles

Зберігає дані про рентну плату, вартість підприємства.

# 4 ТЕСТУВАННЯ ТА ОЦІНКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

## 4.1 Основи тестування програмного забезпечення

* Тестування програмного забезпечення — техніка контролю якості, що перевіряє відповідність між реальною та очікуваною поведінкою програми завдяки кінцевому набору тестів, які обираються певним чином.
* Тестування програмного забезпечення – це процес технічного дослі-дження, призначений для виявлення інформації про якість продукту відносно контексту, в якому він має використовуватись. Техніка тестування також вклю-чає як процес пошуку помилок або інших дефектів, так і випробування програмних складових з метою оцінки. Може оцінюватись:
* відповідність вимогам, якими керувалися проектувальники
* та розробники;
* правильна відповідь для усіх можливих вхідних даних;
* виконання функцій за прийнятний час;
* практичність;
* сумісність з програмним забезпеченням ;
* та операційними системами;
* відповідність задачам замовника.

Оскільки число можливих тестів навіть для нескладних програмних компонент практично нескінченне, тому стратегія тестування полягає в тому, щоб провести всі можливі тести з урахуванням наявного часу та ресурсів. Як результат програмне забезпечення (ПЗ) тестується стандартним виконанням програми з метою виявлення багів (помилок або інших дефектів).

## 4.2 Особливості тестування ігор

У сучасному світі ігрова індустрія зайняла стабільну нішу в сфері світо-вої економіки. Якщо на початковому етапі розвитку цієї сфери вартість розробки гри була не дуже високою, то сьогодні для створення ігор потрібний великий штат розробників, тестувальників (люди, що тестують програмне забезпечення), дизайнерів і т.д.

Стабільний попит на комп'ютерні ігри стимулює інвесторів робити вкладення в розвиток даної сфери. Однак не варто забувати про те, що інвестування в гри може бути не таким успішним, як інвестування в спеціальне програмне забезпечення.

Розробивши ПО, яке повністю задовольняє вимоги користувача і пере-вершує своїх конкурентів на ринку, можна з великою часткою ймовірності очікувати, що це буде економічно вигідне вкладення.

Ігри ж не вирішують бізнес-завдання. Основна функція ігор - розважа-льна. Кінцевий користувач очікує одержати не чітку реалізацію специфікації або іншої документації, а захоплюючий ігровий процес.

Ступінь захопливості гри найчастіше оцінюється протягом перших го-дин знайомства з грою. Якщо перше враження було негативним, гравець навряд чи набуде наступні версії гри, хоч би успішними вони не були.

Тому тестувальник (людина, що тестує програмне забезпечення) про-грамного забезпечення повинен провести тестування ігор до того, як вони на-дійдуть у продаж. Успіх гри може залежати від наступних факторів: механіка гри (властивості програмної моделі, втілені за допомогою анімації і програму-вання), ігровий інтерфейс (сукупність елементів і методів, за допомогою яких користувач управляє грою) і ігровий процес (різні аспекти гри: графіка, звук, дії, які повинен виконати користувач для досягнення мети гри і т.д.).

Існують наступні види тестування ігор:

* Функціональне тестування. Мета - виявити відхилення від функціо-нальних вимог. Зводиться до багаторазового проходження гри, виявлення неполадок і умов, в яких їх можна виправити.
* Навантажувальне тестування. При тестуванні ігор доцільно створити ситуації, які вимагають великої обчислювальної навантаження. Таким чином тестувальник може перевірити продуктивність системи в стресовій ситуації. Так легше помітити і вчасно виправити потенційно ненадійні ділянки коду.
* Тестування локалізації. Ігри часто переводять на мови тих країн, в яких їх припускають випускати на ринок. Трапляється, що перекла-дачі не можуть надати абсолютно точний переклад, який би повністю відповідав подіям гри. Навіть правильно перекладена фраза може не відповідати ситуації і різати слух носію мови. Тому після локалізації корисно провести тестування гри жителями тих країн, де передбачається реалізація кінцевого продукту.

- Тестування на сумісність. Найчастіше програмування ігор проводять на персональних комп'ютерах або ноутбуках. Однак багато ігор мо-жуть бути призначені для інших пристроїв: ігрові приставки, мобільні телефони, комунікатори і т.д. Розробка здійснюється на симуляторах даних пристроїв, проте вони можуть значно відрізнятися від оригіналу. Тому в подальшому можуть виникнути труднощі при запуску гри на оригінальному пристрої. Крім того, слід звернути особливу увагу на ліцензування програмного забезпечення.

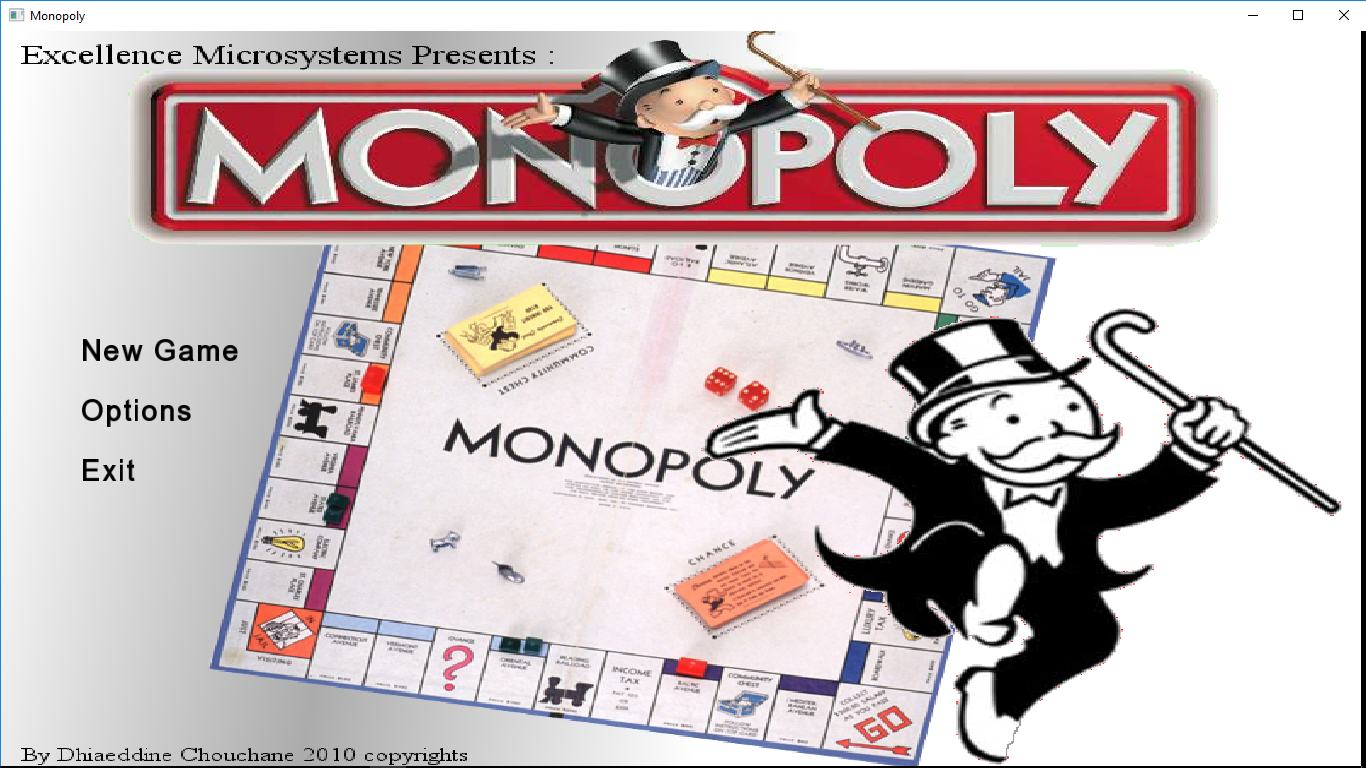
## 4.3 Тестування та результат виконання програми

Тестування програмного продукту у вигляді гри буде проводитись за допомогою функціонального виду тестування, який застосовується виключно для ігор.

Функціональне тестування – один з основних видів незалежного тестування програмного забезпечення, спрямований на перевірку реалізованості функціо-нальних вимог. Інакше кажучи, фахівці з функціонального тестування визна-чають, чи вирішує розроблене ПЗ завдання, заради якого воно було створено, чи задовольняє воно потреби замовника / користувача.

Переваги функціонального тестування:

* імітує фактичне використання системи.
* Недоліки функціонального тестування:
* можливість упущення логічних помилок в програмному забезпеченні;
* ймовірність надмірного тестування.

 Отже, почнемо:

При запуску гри виводиться меню, рисунок 4.1

Рисунок 4.1

При виборі гравців та фігурок, неможливо запустит игру, якщо вибрали тільки одного гравця, це продемонстровано на рисунку 4.2

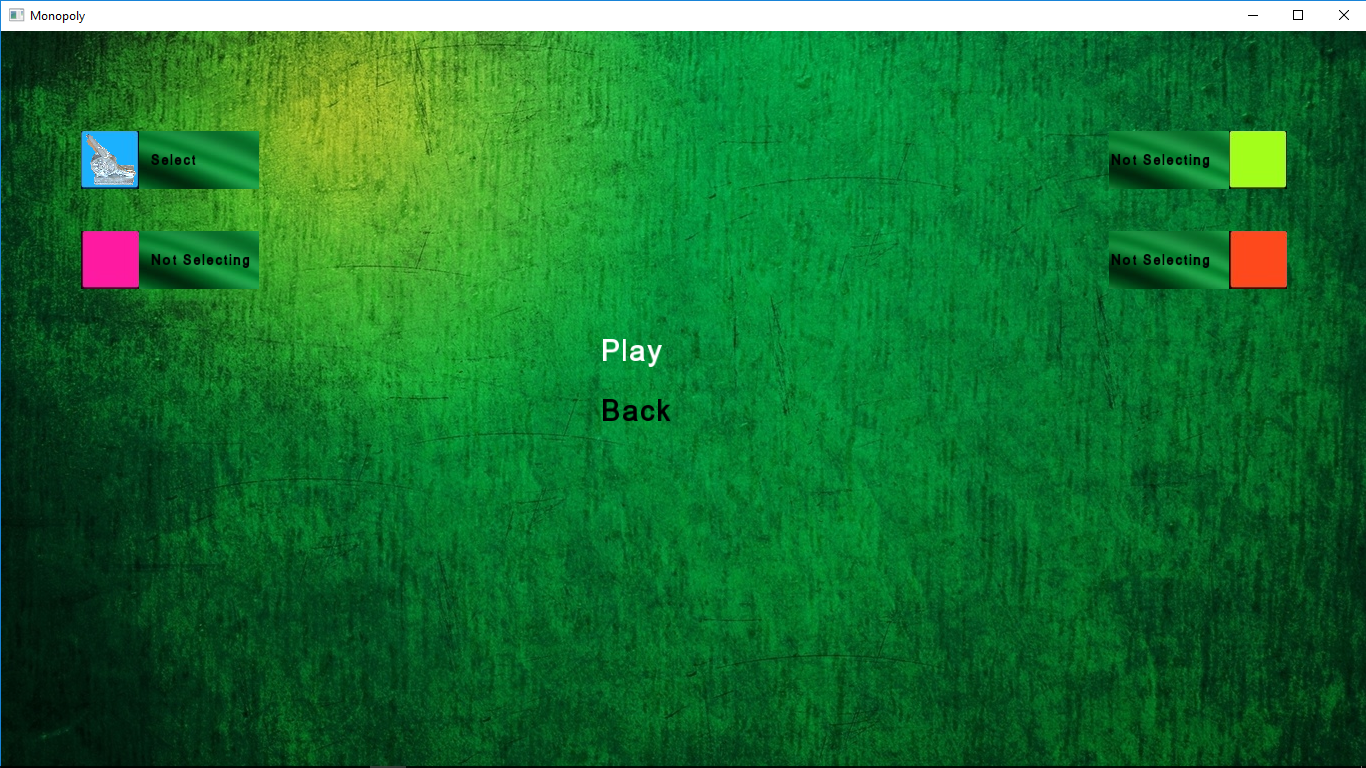


Рисунок 4.2

# ВИСНОВКИ

У ході виконання курсової роботи була створена комп’ютерна гра монополія. Програма дозволяє проявляти свої підприємницькі здібності, співпрацювати з іншими гравцями. Під час етапу проектування була застосована об’єктно-орієнтована парадигм, яка значно полегшила і прискорила етап розробку програмного продукту. Мова С++ дала необхідну швидкодію і потужний інструментарій для реалізації даного ПЗ. Застосування мультимедійної бібліотеки SFML дало необхідний інтерфейс для використання програми.

Визначено основні варіанти використання інформаційної системи, проаналізовано досліджувану проблему, постановлено чітке технічне завдання, спроектовано класи програми, спроектовано можливі варіанти використання, організовано класи програми.

Розробка даного програмного забезпечення здійснювалась з використанням середовища програмування MS Visual Studio Community 201

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Simple and Fast Multimedia Library [Електронний ресурс] // 2.4.2. – 2017. – Режим доступу до ресурсу: <https://www.sfml-dev.org/>.

2. Pupius R. Mastering SFML Game Development Kindle Edition / Raimondas Pupius. – Birmingham: Packt Publishing, 2017. – 442 с.

3. Milchev M. SFML Essentials / Milcho G. Milchev. – Birmingham: Packt Publishing, 2015. – 156 с.

4. Maxime B. SFML Blueprints / Barbier Maxime. – Birmingham: Packt Publishing, 2015. – 298 с.

5. Raimondas P. SFML Game Development By Example / Pupius Raimondas. – Birmingham: Packt Publishing, 2015. – 522 с.

6. Monopoly [Електронний ресурс]. – 2017. – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Monopoly\_(game).

# 

# ДОДАТКИ

# ДОДАТОК A

Лістинг програмного коду

File CoordGo.h

#pragma once

struct CoordGo{

int x, y;

};

File DescriptionOfRectangles.h

#pragma once

struct DescriptionOfRectangles {

int price;

int \*rent;

bool buy;

};

File Coordinate.h

#pragma once

#include<iostream>

using namespace std;

struct Coordinate

{

int x, y;

int Wight, Height;

int leftPx, topPx;

double vec2f1, vec2f2;

bool operator < (const Coordinate &point) const;

bool operator == (const Coordinate &point) const;

};

File Coordinate.cpp

#include "Coordinate.h"

bool Coordinate :: operator < (const Coordinate &point) const {

if (x < point.x)

return true;

return false;

}

bool Coordinate :: operator == (const Coordinate &point) const {

if (this->x == point.x && this->y == point.y && this->Wight == point.Wight && this->Height == point.Height)

return true;

return false;

}

File Chance.h

#pragma once

#include<stdlib.h>

#include<time.h>

#include<iostream>

#include<vector>

using namespace std;

class Chance {

private:

int numChance;

vector<int> isntCheck;

vector<int> isCheck;

vector<int>::iterator it;

public:

Chance();

~Chance();

void setChance();

int getChance(int&, int&);

};

File Chance.cpp

#include "Chance.h"

Chance::Chance(){ }

Chance::~Chance(){}

void Chance::setChance() { for (int i = 0; i < 8; i++) isntCheck.push\_back(i); }

int Chance::getChance(int &pos, int &money) {

srand(time(NULL));

bool flag = false;

int k = 0;

if (isCheck.size() == 8) { isCheck.clear(); setChance();}

while (!flag) {

numChance = rand() % 8;

for (it = isntCheck.begin(); it != isntCheck.end(); it++) {

if (\*it == numChance) {

isCheck.push\_back(\*it);

it = isntCheck.erase(it);

if (it == isntCheck.end()) { k = 1; break; }

flag = true;

break;

}}

if (k == 1) flag = true;

if (isntCheck.empty()) flag = true;

}

if (numChance == 0) pos = 11;

if (numChance == 1) pos = 39;

if (numChance == 2) pos = 24;

if (numChance == 3) money = 150;

if (numChance == 4) pos = 40;

if (numChance == 5) money = -150;

if (numChance == 6) pos = 5;

if (numChance == 7) money = 100;

return numChance;

}

File CommunityChest.h

#pragma once

#include<stdlib.h>

#include<time.h>

#include<iostream>

#include<vector>

using namespace std;

class CommunityChest {

private:

int numComm;

vector<int> isntComm;

vector<int> isComm;

vector<int>::iterator it;

public:

CommunityChest();

~CommunityChest();

void setComm();

int getComm(int&);

};

File CommunityChest.cpp

#include "CommunityChest.h"

CommunityChest::CommunityChest(){}

CommunityChest::~CommunityChest(){}

void CommunityChest::setComm() { for (int i = 0; i < 8; i++) isntComm.push\_back(i);}

int CommunityChest::getComm(int &money) {

srand(time(NULL));

bool flag = false;

int k = 0;

if (isComm.size() == 8) {

isComm.clear();

setComm();}

while (!flag) {

numComm = rand() % 8;

for (it = isntComm.begin(); it != isntComm.end(); it++) {

if (\*it == numComm) {

isComm.push\_back(\*it);

it = isntComm.erase(it);

if (it == isntComm.end()) { k = 1; break; }

flag = true;

break;

}}

if (k == 1) flag = true;

if (isntComm.empty()) flag = true;

}

if (numComm == 0) money = 50;

if (numComm == 1) money = -100;

if (numComm == 2) money = 25;

if (numComm == 3) money = -50;

if (numComm == 4) money = 200;

if (numComm == 5) money = 20;

if (numComm == 6) money = 45;

if (numComm == 7) money = 100;

return numComm;

}

File Dice.h

#pragma once

class Dice {

private:

int \*points;

public:

Dice();

~Dice();

void ThrowDice(int\*);

};

File Dice.cpp

#include "Dice.h"

#include<stdlib.h>

#include<time.h>

Dice::Dice() { }

Dice::~Dice() { }

void Dice::ThrowDice(int\* point) {

points = new int[2];

srand(time(NULL));

for (int i = 0; i < 2; i++) {

points[i] = rand() % 6 + 1; point[i] = points[i]; }

}

File Map.h

#pragma once

#include<iostream>

#include<fstream>

#include<vector>

#include<map>

#include"Coordinate.h"

#include"DescriptionOfRectangles.h"

#include"CoordGo.h"

const int SIZE = 40;

const int FOUR = 4;

using namespace std;

class Map : public DescriptionOfRectangles{

protected:

vector<DescriptionOfRectangles> mapi;

CoordGo go[FOUR][SIZE];

public:

Map();

~Map();

friend ifstream& operator >>(ifstream &, Map &);

void setPosition(int\*, int\*, int, int\*, int&);

void setStartPosition(int\*, int\*, int);

void setBuyerFirm();

bool isBuy(int);

void GetMap(int, long&);

void goToJail(int\*, int\*, int);

int getInfo(int, int);};

File Map.cpp

#include "Map.h"

#include<fstream>

Map::Map() {}

Map::~Map() { }

ifstream& operator >> (ifstream &ifs, Map &obj)

{

for (int i = 0; i < SIZE; i++) {

obj.rent = new int[2];

ifs >> obj.price >> obj.rent[0] >> obj.rent[1];

obj.buy = false;

obj.mapi.push\_back(obj);

}

for (int i = 0; i < FOUR; i++)

for (int j = 0; j < SIZE; j++) {

ifs >> obj.go[i][j].x;

ifs >> obj.go[i][j].y;

}

return ifs;

}

void Map::setPosition(int\* a, int\* b, int c, int \*d, int &point) {

for (int i = 0; i < SIZE; i++) {

if (i == \*d) {

if (point < 40 && \*d < point) { ++i;

a[0] = go[c][i].x;

b[0] = go[c][i].y;

\*d = i;

break;}

else if (i < 39) {

++i;

a[0] = go[c][i].x;

b[0] = go[c][i].y;

\*d = i;

break;}

else {

i = 0;

point %= 40;

a[0] = go[c][i].x;

b[0] = go[c][i].y;

\*d = i;}

}}}

void Map::setStartPosition(int\* a, int\* b, int c) {

a[0] = go[c][0].x;

b[0] = go[c][0].y;

}

bool Map::isBuy(int pos) {

return mapi[pos].buy;

}

void Map::GetMap(int pos, long &money) {

mapi[pos].buy = true;

money -= mapi[pos].price;

}

void Map::goToJail(int\* a, int\* b, int c) {

a[0] = go[c][10].x;

b[0] = go[c][10].y;

}

int Map::getInfo(int pos, int num) {

return mapi[pos].rent[num];

}

File Menu.h

#pragma once

#include <SFML/Graphics.hpp>

#include <string>

#include <fstream>

#include "Tokens.h"

#include"Player.h"

#include<map>

#include<sstream>

#include"Map.h"

using namespace sf;

using namespace std;

class Menu {

private:

Coordinate \*DataTokens;

Texture \*tex\_image, \*tex\_helpInstrument, tex\_Background, \*tex\_check, \*tex\_check1;

Sprite \*spr\_image, \*spr\_helpInstrument, spr\_Background, \*spr\_check, \*spr\_check1;

public:

Menu();

Menu(RenderWindow &);

~Menu();

void NewMenu(RenderWindow &);

void GetDataTokens(int\*, int\*, int\*, int\*, double\*, double\*);

void setImage();

string toStr(int);

void trade(RenderWindow &, Player\*, Map&, int, int, int\*, vector<Sprite>, Text\*);

};

File Menu.cpp

#include "Menu.h"

Menu::Menu() { }

Menu::~Menu() { }

string Menu :: toStr(int var) { stringstream tmp; tmp << var; return tmp.str(); }

Menu::Menu(RenderWindow& window) {

Font fnt\_Arial;

fnt\_Arial.loadFromFile("fonts/arial.ttf");

Text text[3];

string NewGame = "New Game";

string Options = "Options";

string Exit = "Exit";

for (int i = 0; i < 3; i++) {

text[i].setFont(fnt\_Arial);

text[i].setCharacterSize(30);

text[i].setStyle(Text::Bold);

text[i].setFillColor(Color::Black);

}

text[0].setString(NewGame);

text[0].setPosition(80, 300);

text[1].setString(Options);

text[1].setPosition(80, 360);

text[2].setString(Exit);

text[2].setPosition(80, 420);

Texture menuBackground;

menuBackground.loadFromFile("images/intro.tga");

Sprite menuBg(menuBackground);

menuBg.setPosition(0, 0);

menuBg.setScale(Vector2f(1.7f, 1.225f));

bool isMenu = 1;

int menuNum = 0;

while (window.isOpen() && isMenu)

{

Event event;

while (window.pollEvent(event))

{

if (event.type == Event::Closed)

window.close();

}

for (int i = 0; i < 3; i++)

text[i].setFillColor(Color::Black);

menuNum = 0;

window.clear();

if (IntRect(80, 300, 300,50).contains(Mouse::getPosition(window))) { text[0].setFillColor(Color::White); menuNum = 1; }

if (IntRect(80, 360, 300,50).contains(Mouse::getPosition(window))) { text[1].setFillColor(Color::Green); menuNum = 2; }

if (IntRect(80, 420, 300,50).contains(Mouse::getPosition(window))) { text[2].setFillColor(Color::Blue); menuNum = 3; }

if (Mouse::isButtonPressed(Mouse::Left))

{

if (menuNum == 1) {

NewMenu(window);

isMenu = false;

}

if (menuNum == 2) { window.display(); while (!Keyboard::isKeyPressed(Keyboard::Escape)); }

if (menuNum == 3) { window.close(); isMenu = false; }

}

window.draw(menuBg);

for (int i = 0; i < 3; i++)

window.draw(text[i]);

window.display();

}

}

void Menu :: NewMenu(RenderWindow& window) {

DataTokens = new Coordinate[4];

Texture tex\_Plcol[8], tex\_Token[4], menuBackground;

menuBackground.loadFromFile("images/back1.png");

for (int i = 0; i < 8; i++) {

if (i < 4) {

tex\_Token[i].loadFromFile("images/tokens.tga");

tex\_Plcol[i].loadFromFile("images/color.png");

}

if (i > 3) tex\_Plcol[i].loadFromFile("images/Back3.png");

}

Sprite spr\_Plcol[8], spr\_Token[4], menuBg(menuBackground);

menuBg.setPosition(0, 0);

for (int i = 0; i < 8; i++) {

if (i < 4) {

spr\_Plcol[i].setTexture(tex\_Plcol[i]);

spr\_Token[i].setTexture(tex\_Token[i]);

}else spr\_Plcol[i].setTexture(tex\_Plcol[i]);

}

for (int i = 0; i < 8; i++) {

if (i == 0) {

spr\_Plcol[i].setTextureRect(IntRect(55, 104, 58, 58));

spr\_Plcol[i].setPosition(80, 100);

}

if (i == 1) {

spr\_Plcol[i].setTextureRect(IntRect(130, 104, 58, 58));

spr\_Plcol[i].setPosition(1228, 100);

}

if (i == 2) {

spr\_Plcol[i].setTextureRect(IntRect(205, 104, 58, 58));

spr\_Plcol[i].setPosition(80, 200);

}

if (i == 3) {

spr\_Plcol[i].setTextureRect(IntRect(281, 104, 58, 58));

spr\_Plcol[i].setPosition(1228, 200);

}

if (i > 3) { spr\_Plcol[i].setTextureRect(IntRect(0, 0, 120, 58));

if (i == 4) spr\_Plcol[i].setPosition(138, 100);

if (i == 5) spr\_Plcol[i].setPosition(1108, 100);

if (i == 6) spr\_Plcol[i].setPosition(138, 200);

if (i == 7) spr\_Plcol[i].setPosition(1108, 200);

}

}

int a[4], b[4], c[4], d[4];

double e[4], f[4];

for (int i = 0; i < 4; i++){

a[i] = 0, b[i] = 0, c[i] = 1, d[i] = 1, e[i] = 0.0, f[i] = 0.0;

}

for (int i = 0; i < 4; i++) {

spr\_Token[i].setTextureRect(IntRect(a[i], b[i], c[i], d[i]));

if (i == 0) spr\_Token[i].setPosition(80, 100); if (i == 1) spr\_Token[i].setPosition(1228, 100);

if (i == 2) spr\_Token[i].setPosition(80, 200);

if (i == 3) spr\_Token[i].setPosition(1228, 200);

}

Tokens tokens;

ifstream ifs;

ifs.open("Tokens.txt", ios::in);

if (!ifs)

return;

ifs >> tokens;

ifs.close();

bool isMenu = 1;

int menuNum = 0, i = 0;

string Select = "Select";

string NotSelect = "Not Selecting";

string Play = "Play";

string Back = "Back";

Font fnt\_Arial;

fnt\_Arial.loadFromFile("fonts/arial.ttf");

Text text[6];

for (int i = 0; i < 6; i++) {

if (i > 3) {

text[i].setFont(fnt\_Arial);

text[i].setCharacterSize(30);

text[i].setStyle(Text::Bold);

if (i == 4) {

text[i].setString(Play);

text[i].setPosition(600, 300);

}

if (i == 5) {

text[i].setString(Back);

text[i].setPosition(600, 360);

}

continue;

}

text[i].setFont(fnt\_Arial);

text[i].setCharacterSize(14);

text[i].setStyle(Text::Bold);

text[i].setString(NotSelect);

}

text[0].setPosition(150, 120);

text[1].setPosition(1110, 120);

text[2].setPosition(150, 220);

text[3].setPosition(1110, 220);

int isSelect[4] = { 0, 0, 0, 0 };

while (window.isOpen() && isMenu)

{

Event event;

while (window.pollEvent(event))

{

if (event.type == Event::Closed)

window.close();

}

menuNum = 0;

window.clear();

for (int i = 0; i < 6; i++)

text[i].setFillColor(Color::Black);

if (IntRect(138, 100, 120, 58).contains(Mouse::getPosition(window)))

{

text[0].setFillColor(Color::White);

if (Mouse::isButtonPressed(Mouse::Left)) {

if (isSelect[0] == 0) {

isSelect[0] = 1;

text[0].setString(Select);

}

else {

isSelect[0] = 0;

text[0].setString(NotSelect);

a[0] = 0, b[0] = 0, c[0] = 1, d[0] = 1, e[0] = 0.0, f[0] = 0.0;

spr\_Token[0].setTextureRect(IntRect(a[0], b[0], c[0], d[0]));

spr\_Token[0].setPosition(80, 100);

}

}

sleep(\*new Time(milliseconds(150)));

}

if (IntRect(1108, 100, 120, 58).contains(Mouse::getPosition(window))) {

text[1].setFillColor(Color::White);

if (Mouse::isButtonPressed(Mouse::Left)) {

if (isSelect[1] == 0) {

isSelect[1] = 1;

text[1].setString(Select);

}

else {

isSelect[1] = 0;

text[1].setString(NotSelect);

a[1] = 0, b[1] = 0, c[1] = 1, d[1] = 1, e[1] = 0.0, f[1] = 0.0;

spr\_Token[1].setTextureRect(IntRect(a[1], b[1], c[1], d[1]));

spr\_Token[1].setPosition(1228, 100);

}

}

sleep(\*new Time(milliseconds(150)));

}

if (IntRect(138, 200, 120, 58).contains(Mouse::getPosition(window))) {

text[2].setFillColor(Color::White);

if(Mouse::isButtonPressed(Mouse::Left)) {

if (isSelect[2] == 0) {

isSelect[2] = 1;

text[2].setString(Select);

}

else {

isSelect[2] = 0;

text[2].setString(NotSelect);

a[2] = 0, b[2] = 0, c[2] = 1, d[2] = 1, e[2] = 0.0, f[2] = 0.0;

spr\_Token[2].setTextureRect(IntRect(a[2], b[2], c[2], d[2]));

spr\_Token[2].setPosition(80, 200);

}

}

sleep(\*new Time(milliseconds(150)));

}

if (IntRect(1108, 200, 120, 58).contains(Mouse::getPosition(window))) {

text[3].setFillColor(Color::White);

if (Mouse::isButtonPressed(Mouse::Left)) {

if (isSelect[3] == 0) {

isSelect[3] = 1;

text[3].setString(Select);

}

else {

isSelect[3] = 0;

text[3].setString(NotSelect);

a[3] = 0, b[3] = 0, c[3] = 1, d[3] = 1, e[3] = 0.0, f[3] = 0.0;

spr\_Token[3].setTextureRect(IntRect(a[3], b[3], c[3], d[3]));

spr\_Token[3].setPosition(1228, 200);

}

}

sleep(\*new Time(milliseconds(150)));

}

if (IntRect(80, 100, 58, 58).contains(Mouse::getPosition(window))) menuNum = 1;

if (IntRect(1228, 100, 58, 58).contains(Mouse::getPosition(window))) menuNum = 2;

if (IntRect(80, 200, 58, 58).contains(Mouse::getPosition(window))) menuNum = 3;

if (IntRect(1228, 200, 58, 58).contains(Mouse::getPosition(window))) menuNum = 4;

if (IntRect(600, 300, 300, 50).contains(Mouse::getPosition(window))) {

text[4].setFillColor(Color::White);

int count = 0, countT = 0;

for (int i = 0; i < 4; i++) {

if (isSelect[i] == 1) count++;

if (a[i] != 0) countT++;

}

if(count >= 2 && countT >= 2) menuNum = 5;

}

if (IntRect(600, 360, 300, 50).contains(Mouse::getPosition(window))) { text[5].setFillColor(Color::White); menuNum = 6; }

if (Mouse::isButtonPressed(Mouse::Left)) {

if (menuNum == 1 && isSelect[0] == 1) {

tokens.getToken(&a[0], &b[0], &c[0], &d[0], &e[0], &f[0]);

spr\_Token[0].setTextureRect(IntRect(a[0], b[0], c[0], d[0]));

spr\_Token[0].setScale(Vector2f(e[0], f[0]));

spr\_Token[0].setPosition(80, 100);

sleep(\*new Time(milliseconds(200)));

}

if (menuNum == 2 && isSelect[1] == 1) {

tokens.getToken(&a[1], &b[1], &c[1], &d[1], &e[1], &f[1]);

spr\_Token[1].setTextureRect(IntRect(a[1], b[1], c[1], d[1]));

spr\_Token[1].setScale(Vector2f(e[1], f[1]));

spr\_Token[1].setPosition(1228, 100);

sleep(\*new Time(milliseconds(200)));

}

if (menuNum == 3 && isSelect[2] == 1) {

tokens.getToken(&a[2], &b[2], &c[2], &d[2], &e[2], &f[2]);

spr\_Token[2].setTextureRect(IntRect(a[2], b[2], c[2], d[2]));

spr\_Token[2].setScale(Vector2f(e[2], f[2]));

spr\_Token[2].setPosition(80, 200);

sleep(\*new Time(milliseconds(200)));

}

if (menuNum == 4 && isSelect[3] == 1) {

tokens.getToken(&a[3], &b[3], &c[3], &d[3], &e[3], &f[3]);

spr\_Token[3].setTextureRect(IntRect(a[3], b[3], c[3], d[3]));

spr\_Token[3].setScale(Vector2f(e[3], f[3]));

spr\_Token[3].setPosition(1228, 200);

sleep(\*new Time(milliseconds(200)));

}

if (menuNum == 5) {

for (int i = 0; i < 4; i++) {

DataTokens[i].x = a[i];

DataTokens[i].y = b[i];

DataTokens[i].Wight = c[i];

DataTokens[i].Height = d[i];

DataTokens[i].vec2f1 = e[i];

DataTokens[i].vec2f2 = f[i];

}

isMenu = false;

}

if (menuNum == 6) window.close();

}

window.draw(menuBg);

for (int i = 0; i < 8; i++) window.draw(spr\_Plcol[i]);

for (int i = 0; i < 4; i++) window.draw(spr\_Token[i]);

for (int i = 0; i < 6; i++) window.draw(text[i]);

window.display();

}

}

void Menu::GetDataTokens(int\* a, int\* b, int\* c, int\* d, double\* e, double\* f) {

for (int i = 0; i < 4; i++) {

a[i] = DataTokens[i].x;

b[i] = DataTokens[i].y;

c[i] = DataTokens[i].Wight;

d[i] = DataTokens[i].Height;

e[i] = DataTokens[i].vec2f1;

f[i] = DataTokens[i].vec2f2;

}

}

void Menu::setImage() {

tex\_Background.loadFromFile("images/backForTrade.png");

tex\_image = new Texture[28];

tex\_helpInstrument = new Texture[6];

tex\_check = new Texture[18];

tex\_check1 = new Texture[18];

for (int i = 0; i < 18; i++) { tex\_check[i].loadFromFile("images/check.png"); tex\_check1[i].loadFromFile("images/check.png"); }

for (int i = 0; i < 28; i++) tex\_image[i].loadFromFile("images/Firm" + toStr(i + 1) + ".png");

for (int i = 0; i < 6; i++)

{

if(i == 0 || i == 1)

tex\_helpInstrument[i].loadFromFile("images/i1.png");

if (i == 2)

tex\_helpInstrument[i].loadFromFile("images/i2.png");

if (i == 3)

tex\_helpInstrument[i].loadFromFile("images/i3.png");

if (i == 4 || i == 5)

tex\_helpInstrument[i].loadFromFile("images/i4.png");

}

spr\_image = new Sprite[28];

spr\_helpInstrument = new Sprite[6];

spr\_check = new Sprite[18];

spr\_check1 = new Sprite[18];

spr\_Background.setTexture(tex\_Background);

spr\_Background.setPosition(0, 0);

spr\_Background.setScale(Vector2f(2.f, 2.f));

for (int i = 0; i < 18; i++) {

spr\_check[i].setTexture(tex\_check[i]); spr\_check[i].setScale(Vector2f(0.1f, 0.1f));

spr\_check1[i].setTexture(tex\_check1[i]); spr\_check1[i].setScale(Vector2f(0.1f, 0.1f));

}

for (int i = 0; i < 28; i++) { spr\_image[i].setTexture(tex\_image[i]); spr\_image[i].setScale(Vector2f(0.31f, 0.4f)); }

for (int i = 0; i < 6; i++) spr\_helpInstrument[i].setTexture(tex\_helpInstrument[i]);

}

void Menu::trade(RenderWindow &window, Player \*player, Map &map, int Size, int CurrentPlayer, int\* Firm, vector<Sprite> spr\_token, Text \*money) {

bool isMenu = 1;

int menuNum = 0, pos = 0, pos1 = 0;

vector<int> numChecks;

vector<int> numChecks1;

vector<int> one;

vector<int> two;

vector<int>::iterator it;

string mon, mon1;

int moneyFirstPl = 0, moneySecondPl = 0;

int x, y, count, count1, x2, y2, count2 = 0, count3 = 0;

Font fnt\_Arial;

fnt\_Arial.loadFromFile("fonts/arial.ttf");

Text text[6];

for (int i = 0; i < 6; i++) {

text[i].setFont(fnt\_Arial);

text[i].setCharacterSize(20);

text[i].setStyle(Text::Bold);

text[i].setFillColor(Color::White);

if (i == 0) {

text[i].setPosition(610, 0);

text[i].setString("This Player");

}

if (i == 1) {

text[i].setPosition(615, 100);

text[i].setString("Trade with");

}

if (i == 2) {

text[i].setPosition(300, 540);

text[i].setString("Click if all good");

}

if (i == 3) {

text[i].setPosition(983, 540);

text[i].setString("Cick if you agree");

}

if (i == 4) {

text[i].setPosition(500, 700);

}

if (i == 5) {

text[i].setPosition(866, 700);

}

}

spr\_helpInstrument[0].setPosition(315, 570);

spr\_helpInstrument[1].setPosition(998, 570);

spr\_helpInstrument[2].setPosition(1060, 570);

spr\_helpInstrument[3].setPosition(630, 530);

spr\_helpInstrument[4].setPosition(906, 570);

spr\_helpInstrument[5].setPosition(400, 570);

int \*arr, \*arr1, size1 = -1, numPlayer = 0, pl = 0;

bool isClick = false, isChecked[18], isChecked1[18], buy = false;

int size = player[CurrentPlayer].getSize();

arr = new int[size];

for (int i = 0; i < 18; i++) { isChecked[i] = false; isChecked1[i] = false; }

Vector2f obj = spr\_token[CurrentPlayer].getScale();

obj.x \*= 4; obj.y \*= 4;

spr\_token[CurrentPlayer].setPosition(630, 25);

spr\_token[CurrentPlayer].setScale(obj);

int x1 = 630, y1 = 70;

for (int i = 0; i < spr\_token.size(); i++) {

if (i != CurrentPlayer) {

obj = spr\_token[i].getScale();

obj.x \*= 4; obj.y \*= 4;

spr\_token[i].setPosition(x1, y1 += 80);

spr\_token[i].setScale(obj);

}

}

player[CurrentPlayer].sortVector();

player[CurrentPlayer].setPositionMyFirm(arr);

for (int i = 0; i < size; i++) {

for (int j = 0; j < 28; j++) {

if (arr[i] == Firm[j]) {

arr[i] = j;

}

}

}

while (window.isOpen() && isMenu)

{

x = -100; y = 0, count = -1;

x2 = 650; y2 = 0, count1 = -1;

Event event;

while (window.pollEvent(event))

{

if (event.type == Event::Closed)

window.close();

}

menuNum = 0;

window.clear();

spr\_helpInstrument[0].setColor(Color::White);

spr\_helpInstrument[1].setColor(Color::White);

spr\_helpInstrument[2].setColor(Color::White);

spr\_helpInstrument[3].setColor(Color::White);

spr\_helpInstrument[4].setColor(Color::White);

spr\_helpInstrument[5].setColor(Color::White);

for (int i = 0; i < spr\_token.size(); i++)

if (i != CurrentPlayer)

spr\_token[i].setColor(Color::White);

if (spr\_token.size() == 2 && buy == false) {

if (IntRect(630, 150, 80, 80).contains(Mouse::getPosition(window))) {

if (CurrentPlayer != 0) {

spr\_token[0].setColor(Color::Blue);

numPlayer = 0;

menuNum = 1;

}

else if (CurrentPlayer != 1) {

spr\_token[1].setColor(Color::Blue);

numPlayer = 1;

menuNum = 1;

}

}

}

if (spr\_token.size() == 3 && buy == false) {

if (IntRect(630, 150, 80, 80).contains(Mouse::getPosition(window))) {

if (CurrentPlayer == 0) {

spr\_token[1].setColor(Color::Blue);

numPlayer = 1;

menuNum = 1;

}

else if (CurrentPlayer == 1) {

spr\_token[0].setColor(Color::Blue);

numPlayer = 0;

menuNum = 1;

}

}

if (IntRect(630, 230, 80, 80).contains(Mouse::getPosition(window))) {

if (CurrentPlayer == 0) {

spr\_token[2].setColor(Color::Blue);

numPlayer = 2;

menuNum = 2;

}

else if (CurrentPlayer == 1) {

spr\_token[2].setColor(Color::Blue);

numPlayer = 2;

menuNum = 2;

}

else if (CurrentPlayer == 2) {

spr\_token[1].setColor(Color::Blue);

numPlayer = 1;

menuNum = 2;

}

}

}

if (spr\_token.size() == 4 && buy == false) {

if (IntRect(630, 150, 80, 80).contains(Mouse::getPosition(window))) {

if (CurrentPlayer == 0) {

spr\_token[1].setColor(Color::Blue);

numPlayer = 1;

menuNum = 1;

}

else if (CurrentPlayer == 1) {

spr\_token[0].setColor(Color::Blue);

numPlayer = 0;

menuNum = 1;

}

}

if (IntRect(630, 230, 80, 80).contains(Mouse::getPosition(window))) {

if (CurrentPlayer == 0) {

spr\_token[2].setColor(Color::Blue);

numPlayer = 2;

menuNum = 2;

}

else if (CurrentPlayer == 1) {

spr\_token[2].setColor(Color::Blue);

numPlayer = 2;

menuNum = 2;

}

else if (CurrentPlayer == 2) {

spr\_token[1].setColor(Color::Blue);

numPlayer = 1;

menuNum = 2;

}

}

if (IntRect(630, 310, 80, 80).contains(Mouse::getPosition(window))) {

if (CurrentPlayer == 0) {

spr\_token[3].setColor(Color::Blue);

numPlayer = 3;

menuNum = 3;

}

else if (CurrentPlayer == 1) {

spr\_token[3].setColor(Color::Blue);

numPlayer = 3;

menuNum = 3;

}

else if (CurrentPlayer == 2) {

spr\_token[3].setColor(Color::Blue);

numPlayer = 3;

menuNum = 3;

}

else if (CurrentPlayer == 3) {

spr\_token[2].setColor(Color::Blue);

numPlayer = 2;

menuNum = 3;

}

}

}

if (IntRect(315, 570, 58, 58).contains(Mouse::getPosition(window))) {

spr\_helpInstrument[0].setColor(Color::Blue);

menuNum = 4;

}

if (IntRect(998, 570, 58, 58).contains(Mouse::getPosition(window))) {

spr\_helpInstrument[1].setColor(Color::Blue);

menuNum = 5;

}

if (IntRect(1060, 570, 58, 58).contains(Mouse::getPosition(window))) {

spr\_helpInstrument[2].setColor(Color::Blue);

menuNum = 6;

}

if (IntRect(630, 530, 65, 63).contains(Mouse::getPosition(window))) {

spr\_helpInstrument[3].setColor(Color::Blue);

menuNum = 7;

}

if (IntRect(906, 570, 74, 33).contains(Mouse::getPosition(window))) {

spr\_helpInstrument[4].setColor(Color::Blue);

menuNum = 8;

}

if (IntRect(400, 570, 74, 33).contains(Mouse::getPosition(window))) {

spr\_helpInstrument[5].setColor(Color::Blue);

menuNum = 9;

}

if (Mouse::isButtonPressed(Mouse::Left)) {

if (menuNum == 1 && buy == false) {

pl = numPlayer;

size1 = player[pl].getSize();

arr1 = new int[size1];

player[pl].sortVector();

player[pl].setPositionMyFirm(arr1);

for (int i = 0; i < size1; i++) {

for (int j = 0; j < 28; j++) {

if (arr1[i] == Firm[j]) {

arr1[i] = j;

}

}

}

isClick = true;

}

if (menuNum == 2 && buy == false) {

pl = numPlayer;

size1 = player[pl].getSize();

arr1 = new int[size1];

player[pl].sortVector();

player[pl].setPositionMyFirm(arr1);

for (int i = 0; i < size1; i++) {

for (int j = 0; j < 28; j++) {

if (arr1[i] == Firm[j]) {

arr1[i] = j;

}

}

}

isClick = true;

}

if (menuNum == 3 && buy == false) {

pl = numPlayer;

size1 = player[pl].getSize();

arr1 = new int[size1];

player[pl].sortVector();

player[pl].setPositionMyFirm(arr1);

for (int i = 0; i < size1; i++) {

for (int j = 0; j < 28; j++) {

if (arr1[i] == Firm[j]) {

arr1[i] = j;

}

}

}

isClick = true;

}

if (menuNum == 4 && (!numChecks.empty() || moneyFirstPl != 0)) {

for (int i = 0; i < numChecks.size(); i++) {

one.push\_back(arr[numChecks[i] - 1]);

}

for (int i = 0; i < numChecks1.size(); i++) {

two.push\_back(arr1[numChecks1[i] - 1]);

}

buy = true;

}

if (menuNum == 5 && (!numChecks.empty() || moneySecondPl != 0) && buy == true) {

player[CurrentPlayer].Change(one, two, Firm, moneySecondPl, moneyFirstPl);

player[pl].Change(two, one, Firm, moneyFirstPl, moneySecondPl);

isMenu = false;

}

if (menuNum == 6 && buy == true) { buy = false; }

if (menuNum == 7 && buy == true) { isMenu = false; }

if (menuNum == 8 && buy == false) {

bool flag = false;

string sTemp;

while (!flag) {

if (Keyboard::isKeyPressed(Keyboard::Num0)) sTemp += '0';

if (Keyboard::isKeyPressed(Keyboard::Num1)) sTemp += '1';

if (Keyboard::isKeyPressed(Keyboard::Num2)) sTemp += '2';

if (Keyboard::isKeyPressed(Keyboard::Num3)) sTemp += '3';

if (Keyboard::isKeyPressed(Keyboard::Num4)) sTemp += '4';

if (Keyboard::isKeyPressed(Keyboard::Num5)) sTemp += '5';

if (Keyboard::isKeyPressed(Keyboard::Num6)) sTemp += '6';

if (Keyboard::isKeyPressed(Keyboard::Num7)) sTemp += '7';

if (Keyboard::isKeyPressed(Keyboard::Num8)) sTemp += '8';

if (Keyboard::isKeyPressed(Keyboard::Num9)) sTemp += '9';

if (Keyboard::isKeyPressed(Keyboard::Space)) flag = true;

sleep(\*new Time(milliseconds(100)));

}

mon1 = sTemp;

istringstream ist(mon1);

ist >> moneySecondPl;

if (moneySecondPl > player[pl].getMoneyPlayer()) {

moneySecondPl = 0;

}

}

if (menuNum == 9 && buy == false) {

bool flag = false;

string sTemp;

while (!flag) {

if (Keyboard::isKeyPressed(Keyboard::Num0)) sTemp += '0';

if (Keyboard::isKeyPressed(Keyboard::Num1)) sTemp += '1';

if (Keyboard::isKeyPressed(Keyboard::Num2)) sTemp += '2';

if (Keyboard::isKeyPressed(Keyboard::Num3)) sTemp += '3';

if (Keyboard::isKeyPressed(Keyboard::Num4)) sTemp += '4';

if (Keyboard::isKeyPressed(Keyboard::Num5)) sTemp += '5';

if (Keyboard::isKeyPressed(Keyboard::Num6)) sTemp += '6';

if (Keyboard::isKeyPressed(Keyboard::Num7)) sTemp += '7';

if (Keyboard::isKeyPressed(Keyboard::Num8)) sTemp += '8';

if (Keyboard::isKeyPressed(Keyboard::Num9)) sTemp += '9';

if (Keyboard::isKeyPressed(Keyboard::Space)) flag = true;

sleep(\*new Time(milliseconds(100)));

}

mon = sTemp;

istringstream ist(mon);

ist >> moneyFirstPl;

if (moneyFirstPl > player[CurrentPlayer].getMoneyPlayer()) {

moneyFirstPl = 0;

}

}

}

if (buy == false) {

if (IntRect(0, 0, 100, 185).contains(Mouse::getPosition(window)) && size > 0) pos = 1;

if (IntRect(100, 0, 100, 185).contains(Mouse::getPosition(window)) && size > 1) pos = 2;

if (IntRect(200, 0, 100, 185).contains(Mouse::getPosition(window)) && size > 2) pos = 3;

if (IntRect(300, 0, 100, 185).contains(Mouse::getPosition(window)) && size > 3) pos = 4;

if (IntRect(400, 0, 100, 185).contains(Mouse::getPosition(window)) && size > 4) pos = 5;

if (IntRect(500, 0, 100, 185).contains(Mouse::getPosition(window)) && size > 5) pos = 6;

if (IntRect(0, 0, 100, 185).contains(Mouse::getPosition(window)) && size > 6) pos = 7;

if (IntRect(100, 187, 100, 185).contains(Mouse::getPosition(window)) && size > 7) pos = 8;

if (IntRect(200, 187, 100, 185).contains(Mouse::getPosition(window)) && size > 8) pos = 9;

if (IntRect(300, 187, 100, 185).contains(Mouse::getPosition(window)) && size > 9) pos = 10;

if (IntRect(400, 187, 100, 185).contains(Mouse::getPosition(window)) && size > 10) pos = 11;

if (IntRect(500, 187, 100, 185).contains(Mouse::getPosition(window)) && size > 11) pos = 12;

if (IntRect(0, 372, 100, 185).contains(Mouse::getPosition(window)) && size > 12) pos = 13;

if (IntRect(100, 372, 100, 185).contains(Mouse::getPosition(window)) && size > 13) pos = 14;

if (IntRect(200, 372, 100, 185).contains(Mouse::getPosition(window)) && size > 14) pos = 15;

if (IntRect(300, 372, 100, 185).contains(Mouse::getPosition(window)) && size > 15) pos = 16;

if (IntRect(400, 372, 100, 185).contains(Mouse::getPosition(window)) && size > 16) pos = 17;

if (IntRect(500, 372, 100, 185).contains(Mouse::getPosition(window)) && size > 17) pos = 18;

if (Mouse::isButtonPressed(Mouse::Left)) {

if (pos == 1 && isChecked[0] == false) {

spr\_check[0].setPosition(0, 0);

numChecks.push\_back(pos);

isChecked[0] = true;

}

else if (pos == 1) {

for (it = numChecks.begin(); it != numChecks.end(); it++) {

if (\*it == pos) it = numChecks.erase(it);

if (it == numChecks.end()) break;

}

isChecked[0] = false;

}

if (pos == 2 && isChecked[1] == false) { spr\_check[1].setPosition(100, 0); numChecks.push\_back(pos); isChecked[1] = true; }

else if (pos == 2) {

for (it = numChecks.begin(); it != numChecks.end(); it++) {

if (\*it == pos) it = numChecks.erase(it);

if (it == numChecks.end()) break;

}

isChecked[1] = false;

}

if (pos == 3 && isChecked[2] == false) { spr\_check[2].setPosition(200, 0); numChecks.push\_back(pos); isChecked[2] = true; }

else if (pos == 3) {

for (it = numChecks.begin(); it != numChecks.end(); it++) {

if (\*it == pos) it = numChecks.erase(it);

if (it == numChecks.end()) break;

}

isChecked[2] = false;

}

if (pos == 4 && isChecked[3] == false) { spr\_check[3].setPosition(300, 0); numChecks.push\_back(pos); isChecked[3] = true; }

else if (pos == 4) {

for (it = numChecks.begin(); it != numChecks.end(); it++) {

if (\*it == pos) it = numChecks.erase(it);

if (it == numChecks.end()) break;

}

isChecked[3] = false;

}

if (pos == 5 && isChecked[4] == false) { spr\_check[4].setPosition(400, 0); numChecks.push\_back(pos); isChecked[4] = true; }

else if (pos == 5) {

for (it = numChecks.begin(); it != numChecks.end(); it++) {

if (\*it == pos) it = numChecks.erase(it);

if (it == numChecks.end()) break;

}

isChecked[4] = false;

}

if (pos == 6 && isChecked[5] == false) { spr\_check[5].setPosition(500, 0); numChecks.push\_back(pos); isChecked[5] = true; }

else if (pos == 6) {

for (it = numChecks.begin(); it != numChecks.end(); it++) {

if (\*it == pos) it = numChecks.erase(it);

if (it == numChecks.end()) break;

}

isChecked[5] = false;

}

if (pos == 7 && isChecked[6] == false) { spr\_check[6].setPosition(0, 187); numChecks.push\_back(pos); isChecked[6] = true; }

else if (pos == 7) {

for (it = numChecks.begin(); it != numChecks.end(); it++) {

if (\*it == pos) it = numChecks.erase(it);

if (it == numChecks.end()) break;

}

isChecked[6] = false;

}

if (pos == 8 && isChecked[7] == false) { spr\_check[7].setPosition(100, 187); numChecks.push\_back(pos); isChecked[7] = true; }

else if (pos == 8) {

for (it = numChecks.begin(); it != numChecks.end(); it++) {

if (\*it == pos) it = numChecks.erase(it);

if (it == numChecks.end()) break;

}

isChecked[7] = false;

}

if (pos == 9 && isChecked[8] == false) { spr\_check[8].setPosition(200, 187); numChecks.push\_back(pos); isChecked[8] = true; }

else if (pos == 9) {

for (it = numChecks.begin(); it != numChecks.end(); it++) {

if (\*it == pos) it = numChecks.erase(it);

if (it == numChecks.end()) break;

}

isChecked[8] = false;

}

if (pos == 10 && isChecked[9] == false) { spr\_check[9].setPosition(300, 187); numChecks.push\_back(pos); isChecked[9] = true; }

else if (pos == 10) {

for (it = numChecks.begin(); it != numChecks.end(); it++) {

if (\*it == pos) it = numChecks.erase(it);

if (it == numChecks.end()) break;

}

isChecked[9] = false;

}

if (pos == 11 && isChecked[10] == false) { spr\_check[10].setPosition(400, 187); numChecks.push\_back(pos); isChecked[10] = true; }

else if (pos == 11) {

for (it = numChecks.begin(); it != numChecks.end(); it++) {

if (\*it == pos) it = numChecks.erase(it);

if (it == numChecks.end()) break;

}

isChecked[10] = false;

}

if (pos == 12 && isChecked[11] == false) { spr\_check[11].setPosition(500, 187); numChecks.push\_back(pos); isChecked[11] = true; }

else if (pos == 12) {

for (it = numChecks.begin(); it != numChecks.end(); it++) {

if (\*it == pos) it = numChecks.erase(it);

if (it == numChecks.end()) break;

}

isChecked[11] = false;

}

if (pos == 13 && isChecked[12] == false) { spr\_check[12].setPosition(0, 372); numChecks.push\_back(pos); isChecked[12] = true; }

else if (pos == 13) {

for (it = numChecks.begin(); it != numChecks.end(); it++) {

if (\*it == pos) it = numChecks.erase(it);

if (it == numChecks.end()) break;

}

isChecked[12] = false;

}

if (pos == 14 && isChecked[13] == false) { spr\_check[13].setPosition(100, 372); numChecks.push\_back(pos); isChecked[13] = true; }

else if (pos == 14) {

for (it = numChecks.begin(); it != numChecks.end(); it++) {

if (\*it == pos) it = numChecks.erase(it);

if (it == numChecks.end()) break;

}

isChecked[13] = false;

}

if (pos == 15 && isChecked[14] == false) { spr\_check[14].setPosition(200, 372); numChecks.push\_back(pos); isChecked[14] = true; }

else if (pos == 15) {

for (it = numChecks.begin(); it != numChecks.end(); it++) {

if (\*it == pos) it = numChecks.erase(it);

if (it == numChecks.end()) break;

}

isChecked[14] = false;

}

if (pos == 16 && isChecked[15] == false) { spr\_check[15].setPosition(300, 372); numChecks.push\_back(pos); isChecked[15] = true; }

else if (pos == 16) {

for (it = numChecks.begin(); it != numChecks.end(); it++) {

if (\*it == pos) it = numChecks.erase(it);

if (it == numChecks.end()) break;

}

isChecked[15] = false;

}

if (pos == 17 && isChecked[16] == false) { spr\_check[16].setPosition(400, 372); numChecks.push\_back(pos); isChecked[16] = true; }

else if (pos == 17) {

for (it = numChecks.begin(); it != numChecks.end(); it++) {

if (\*it == pos) it = numChecks.erase(it);

if (it == numChecks.end()) break;

}

isChecked[16] = false;

}

if (pos == 18 && isChecked[17] == false) { spr\_check[17].setPosition(500, 372); numChecks.push\_back(pos); isChecked[17] = true; }

else if (pos == 18) {

for (it = numChecks.begin(); it != numChecks.end(); it++) {

if (\*it == pos) it = numChecks.erase(it);

if (it == numChecks.end()) break;

}

isChecked[17] = false;

}

sleep(\*new Time(milliseconds(150)));

}

}

if (buy == false) {

if (IntRect(750, 0, 100, 185).contains(Mouse::getPosition(window)) && size1 > 0) pos1 = 1;

if (IntRect(850, 0, 100, 185).contains(Mouse::getPosition(window)) && size1 > 1) pos1 = 2;

if (IntRect(950, 0, 100, 185).contains(Mouse::getPosition(window)) && size1 > 2) pos1 = 3;

if (IntRect(1050, 0, 100, 185).contains(Mouse::getPosition(window)) && size1 > 3) pos1 = 4;

if (IntRect(1150, 0, 100, 185).contains(Mouse::getPosition(window)) && size1 > 4) pos1 = 5;

if (IntRect(1250, 0, 100, 185).contains(Mouse::getPosition(window)) && size1 > 5) pos1 = 6;

if (IntRect(750, 0, 100, 185).contains(Mouse::getPosition(window)) && size1 > 6) pos1 = 7;

if (IntRect(850, 187, 100, 185).contains(Mouse::getPosition(window)) && size1 > 7) pos1 = 8;

if (IntRect(950, 187, 100, 185).contains(Mouse::getPosition(window)) && size1 > 8) pos1 = 9;

if (IntRect(1050, 187, 100, 185).contains(Mouse::getPosition(window)) && size1 > 9) pos1 = 10;

if (IntRect(1150, 187, 100, 185).contains(Mouse::getPosition(window)) && size1 > 10) pos1 = 11;

if (IntRect(1250, 187, 100, 185).contains(Mouse::getPosition(window)) && size1 > 11) pos1 = 12;

if (IntRect(750, 372, 100, 185).contains(Mouse::getPosition(window)) && size1 > 12) pos1 = 13;

if (IntRect(850, 372, 100, 185).contains(Mouse::getPosition(window)) && size1 > 13) pos1 = 14;

if (IntRect(950, 372, 100, 185).contains(Mouse::getPosition(window)) && size1 > 14) pos1 = 15;

if (IntRect(1050, 372, 100, 185).contains(Mouse::getPosition(window)) && size1 > 15) pos1 = 16;

if (IntRect(1150, 372, 100, 185).contains(Mouse::getPosition(window)) && size1 > 16) pos1 = 17;

if (IntRect(1250, 372, 100, 185).contains(Mouse::getPosition(window)) && size1 > 17) pos1 = 18;

if (Mouse::isButtonPressed(Mouse::Left) && isClick == true) {

if (pos1 == 1 && isChecked1[0] == false) {

spr\_check1[0].setPosition(750, 0);

numChecks1.push\_back(pos1);

isChecked1[0] = true;

}

else if (pos1 == 1) {

for (it = numChecks1.begin(); it != numChecks1.end(); it++) {

if (\*it == pos1) it = numChecks1.erase(it);

if (it == numChecks1.end()) break;

}

isChecked1[0] = false;

}

if (pos1 == 2 && isChecked1[1] == false) {

spr\_check1[1].setPosition(850, 0);

numChecks1.push\_back(pos1);

isChecked1[1] = true;

}

else if (pos1 == 2) {

for (it = numChecks1.begin(); it != numChecks1.end(); it++) {

if (\*it == pos1) it = numChecks1.erase(it);

if (it == numChecks1.end()) break;

}

isChecked1[1] = false;

}

if (pos1 == 3 && isChecked1[2] == false) { spr\_check1[2].setPosition(950, 0); numChecks1.push\_back(pos1); isChecked1[2] = true; }

else if (pos1 == 21) {

for (it = numChecks1.begin(); it != numChecks1.end(); it++) {

if (\*it == pos1) it = numChecks1.erase(it);

if (it == numChecks1.end()) break;

}

isChecked[2] = false;

}

if (pos1 == 4 && isChecked1[3] == false) { spr\_check1[3].setPosition(950, 0); numChecks1.push\_back(pos1); isChecked1[3] = true; }

else if (pos1 == 4) {

for (it = numChecks1.begin(); it != numChecks1.end(); it++) {

if (\*it == pos1) it = numChecks1.erase(it);

if (it == numChecks1.end()) break;

}

isChecked[3] = false;

}

if (pos1 == 5 && isChecked1[4] == false) { spr\_check1[4].setPosition(950, 0); numChecks1.push\_back(pos1); isChecked1[4] = true; }

else if (pos1 == 5) {

for (it = numChecks1.begin(); it != numChecks1.end(); it++) {

if (\*it == pos1) it = numChecks1.erase(it);

if (it == numChecks1.end()) break;

}

isChecked[5] = false;

}

if (pos1 == 6 && isChecked1[5] == false) { spr\_check1[5].setPosition(950, 0); numChecks1.push\_back(pos1); isChecked1[5] = true; }

else if (pos1 == 6) {

for (it = numChecks1.begin(); it != numChecks1.end(); it++) {

if (\*it == pos1) it = numChecks1.erase(it);

if (it == numChecks1.end()) break;

}

isChecked[5] = false;

}

if (pos1 == 7 && isChecked1[6] == false) { spr\_check1[6].setPosition(950, 0); numChecks1.push\_back(pos1); isChecked1[6] = true; }

else if (pos1 == 7) {

for (it = numChecks1.begin(); it != numChecks1.end(); it++) {

if (\*it == pos1) it = numChecks1.erase(it);

if (it == numChecks1.end()) break;

}

isChecked[6] = false;

}

if (pos1 == 8 && isChecked1[7] == false) { spr\_check1[7].setPosition(950, 0); numChecks1.push\_back(pos1); isChecked1[7] = true; }

else if (pos1 == 8) {

for (it = numChecks1.begin(); it != numChecks1.end(); it++) {

if (\*it == pos1) it = numChecks1.erase(it);

if (it == numChecks1.end()) break;

}

isChecked[7] = false;

}

if (pos1 == 9 && isChecked1[8] == false) { spr\_check1[8].setPosition(950, 0); numChecks1.push\_back(pos1); isChecked1[8] = true; }

else if (pos1 == 9) {

for (it = numChecks1.begin(); it != numChecks1.end(); it++) {

if (\*it == pos1) it = numChecks1.erase(it);

if (it == numChecks1.end()) break;

}

isChecked[8] = false;

}

if (pos1 == 10 && isChecked1[9] == false) { spr\_check1[9].setPosition(950, 0); numChecks1.push\_back(pos1); isChecked1[9] = true; }

else if (pos1 == 10) {

for (it = numChecks1.begin(); it != numChecks1.end(); it++) {

if (\*it == pos1) it = numChecks1.erase(it);

if (it == numChecks1.end()) break;

}

isChecked[9] = false;

}

if (pos1 == 11 && isChecked1[10] == false) { spr\_check1[10].setPosition(950, 0); numChecks1.push\_back(pos1); isChecked1[10] = true; }

else if (pos1 == 11) {

for (it = numChecks1.begin(); it != numChecks1.end(); it++) {

if (\*it == pos1) it = numChecks1.erase(it);

if (it == numChecks1.end()) break;

}

isChecked[10] = false;

}

if (pos1 == 12 && isChecked1[11] == false) { spr\_check1[11].setPosition(950, 0); numChecks1.push\_back(pos1); isChecked1[11] = true; }

else if (pos1 == 12) {

for (it = numChecks1.begin(); it != numChecks1.end(); it++) {

if (\*it == pos1) it = numChecks1.erase(it);

if (it == numChecks1.end()) break;

}

isChecked[11] = false;

}

if (pos1 == 13 && isChecked1[12] == false) { spr\_check1[12].setPosition(950, 0); numChecks1.push\_back(pos1); isChecked1[12] = true; }

else if (pos1 == 13) {

for (it = numChecks1.begin(); it != numChecks1.end(); it++) {

if (\*it == pos1) it = numChecks1.erase(it);

if (it == numChecks1.end()) break;

}

isChecked[12] = false;

}

if (pos1 == 14 && isChecked1[13] == false) { spr\_check1[13].setPosition(950, 0); numChecks1.push\_back(pos1); isChecked1[13] = true; }

else if (pos1 == 14) {

for (it = numChecks1.begin(); it != numChecks1.end(); it++) {

if (\*it == pos1) it = numChecks1.erase(it);

if (it == numChecks1.end()) break;

}

isChecked[13] = false;

}

if (pos1 == 15 && isChecked1[14] == false) { spr\_check1[14].setPosition(950, 0); numChecks1.push\_back(pos1); isChecked1[14] = true; }

else if (pos1 == 15) {

for (it = numChecks1.begin(); it != numChecks1.end(); it++) {

if (\*it == pos1) it = numChecks1.erase(it);

if (it == numChecks1.end()) break;

}

isChecked[14] = false;

}

if (pos1 == 16 && isChecked1[15] == false) { spr\_check1[15].setPosition(950, 0); numChecks1.push\_back(pos1); isChecked1[15] = true; }

else if (pos1 == 16) {

for (it = numChecks1.begin(); it != numChecks1.end(); it++) {

if (\*it == pos1) it = numChecks1.erase(it);

if (it == numChecks1.end()) break;

}

isChecked[15] = false;

}

if (pos1 == 17 && isChecked1[16] == false) { spr\_check1[16].setPosition(950, 0); numChecks1.push\_back(pos1); isChecked1[16] = true; }

else if (pos1 == 17) {

for (it = numChecks1.begin(); it != numChecks1.end(); it++) {

if (\*it == pos1) it = numChecks1.erase(it);

if (it == numChecks1.end()) break;

}

isChecked[16] = false;

}

if (pos1 == 18 && isChecked1[17] == false) { spr\_check1[17].setPosition(950, 0); numChecks1.push\_back(pos1); isChecked1[17] = true; }

else if (pos1 == 18) {

for (it = numChecks1.begin(); it != numChecks1.end(); it++) {

if (\*it == pos1) it = numChecks1.erase(it);

if (it == numChecks1.end()) break;

}

isChecked[17] = false;

}

sleep(\*new Time(milliseconds(150)));

}

}

window.draw(spr\_Background);

if (isClick == true) {

for (int i = 0; i < size1; i++) {

++count1;

if (count1 % 6 == 0 && count1 != 0) {

y2 += 187;

x2 = 600;

}

spr\_image[arr1[i]].setPosition(x2 += 100, y2);

window.draw(spr\_image[arr1[i]]);

}

}

for (int i = 0; i < size; i++) {

++count;

if (count % 6 == 0 && count != 0) {

y += 187;

x = -100;

}

spr\_image[arr[i]].setPosition(x += 100, y);

window.draw(spr\_image[arr[i]]);

}

text[4].setString(to\_string(moneyFirstPl));

text[5].setString(to\_string(moneySecondPl));

for (int i = 0; i < 6; i++) window.draw(text[i]);

for(int i = 0; i < spr\_token.size(); i++) window.draw(spr\_token[i]);

for (int i = 0; i < numChecks.size(); i++) {

window.draw(spr\_check[numChecks[i] - 1]);

}

for (int i = 0; i < numChecks1.size(); i++) {

window.draw(spr\_check1[numChecks1[i] - 1]);

}

for (int i = 0; i < 6; i++) {

window.draw(spr\_helpInstrument[i]);

}

window.display();

sleep(\*new Time(milliseconds(150)));

}

}

File Mono.h

#pragma once

#include<SFML/Graphics.hpp>

#include<iostream>

#include<fstream>

#include<string>

#include<algorithm>

#include<list>

#include<sstream>

#include<map>

#include"Menu.h"

#include"Map.h"

#include"Dice.h"

#include"Chance.h"

#include"CommunityChest.h"

#include"Player.h"

#include<vector>

using namespace sf;

using namespace std;

const int ZERO = 0;

const int ONE = 1;

const int TWO = 2;

const int FIVE = 5;

const int SIX = 6;

const int EIGHT = 8;

const int TWEIGHT = 28;

class Mono

{

private:

int left, top, menuNum, Size, nextPosition, pos, posForBuy, j, Numplayer;

int \*point, \*count, \*PlayerInJail, \*a, \*b, \*c, \*d, \*g, \*h, \*control;

double \*e, \*f;

long PlayerMoney;

bool fl, isRoolDice, go, buy;

Font fnt\_Arial;

RenderWindow App;

Text text, \*money;

Texture txt\_map, tex\_tokens, tex\_background, \*tex\_PlayerZone, \*tex\_TokenInPlayerZone;

Texture \*tex\_Token, \*tex\_Dice, \*tex\_Button, \*tex\_Rect, \*tex\_Chance, \*tex\_Comm;

Sprite spr\_backgr, spr\_map, \*spr\_Dice, \*spr\_Button, \*spr\_Rect;

Sprite \*spr\_Chance, \*spr\_Comm, \*spr\_PlayerZone, \*spr\_TokenInPlayerZone;

ifstream ifs;

Map map;

Player \*player;

Dice dice;

public:

Mono();

~Mono();

void start();

string toStr(int);

};

File Mono.cpp

#include "Mono.h"

Mono::Mono(){}

Mono::~Mono(){ }

void Mono::start() {

App.create(VideoMode::getDesktopMode(), "Monopoly");

point = new int[TWO];

control = new int[TWO];

a = new int[FOUR];

b = new int[FOUR];

c = new int[FOUR];

d = new int[FOUR];

g = new int[FOUR];

h = new int[FOUR];

e = new double[FOUR];

f = new double[FOUR];

tex\_Token = new Texture[FOUR];

tex\_Dice = new Texture[SIX];

tex\_Button = new Texture[FIVE];

tex\_Rect = new Texture[TWEIGHT];

tex\_Chance = new Texture[EIGHT];

tex\_Comm = new Texture[EIGHT];

spr\_Dice = new Sprite[SIX];

spr\_Button = new Sprite[FIVE];

spr\_Rect = new Sprite[TWEIGHT];

spr\_Chance = new Sprite[EIGHT];

spr\_Comm = new Sprite[EIGHT];

left = 300, top = 0, menuNum = 0;

fnt\_Arial.loadFromFile("fonts/arial.ttf");

text.setFont(fnt\_Arial);

text.setCharacterSize(20);

text.setStyle(Text::Bold);

txt\_map.loadFromFile("images/map.png");

tex\_tokens.loadFromFile("images/tokens.tga");

tex\_background.loadFromFile("images/back.png");

for (int i = 0; i < 28; i++) tex\_Rect[i].loadFromFile("images/white.png");

for (int i = 0; i < 5; i++) tex\_Button[i].loadFromFile("images/button" + toStr(i + 1) + ".png");

for (int i = 0; i < 4; i++) tex\_Token[i].loadFromFile("images/tokens.tga");

for (int i = 0; i < 6; i++) tex\_Dice[i].loadFromFile("images/de" + toStr(i + 1) + ".png");

for (int i = 0; i < 8; i++) {

tex\_Chance[i].loadFromFile("images/chance/" + toStr(i + 1) + ".png");

tex\_Comm[i].loadFromFile("images/chance/c" + toStr(i + 1) + ".png"); }

vector<Sprite> spr\_Token(4);

spr\_backgr.setTexture(tex\_background);

spr\_map.setTexture(txt\_map);

spr\_backgr.setPosition(0, 0);

spr\_map.setScale(Vector2f(0.45f, 0.45f));

spr\_map.setPosition(385, 6);

for (int i = 0; i < 8; i++) {

spr\_Chance[i].setTexture(tex\_Chance[i]);

spr\_Chance[i].setPosition(455, 259);

spr\_Comm[i].setTexture(tex\_Comm[i]);

spr\_Comm[i].setPosition(455, 259);

}

for (int i = 0; i < 28; i++) {

spr\_Rect[i].setTexture(tex\_Rect[i]);

if ((i >= 0 && i < 6) || (i > 13 && i < 22)) {

spr\_Rect[i].setTextureRect(IntRect(1, 1, 41, 10));

}

else {

spr\_Rect[i].setTextureRect(IntRect(1, 1, 10, 41));

}

}

for (int i = 0; i < 28; i++) {

switch (i) {

case 0: spr\_Rect[i].setPosition(925, 695); break;

case 1: spr\_Rect[i].setPosition(814, 695); break;

case 2: spr\_Rect[i].setPosition(703, 695); break;

case 3: spr\_Rect[i].setPosition(645, 695); break;

case 4: spr\_Rect[i].setPosition(537, 695); break;

case 5: spr\_Rect[i].setPosition(480, 695); break;

case 6: spr\_Rect[i].setPosition(370, 545); break;

case 7: spr\_Rect[i].setPosition(370, 488); break;

case 8: spr\_Rect[i].setPosition(370, 431); break;

case 9: spr\_Rect[i].setPosition(370, 377); break;

case 10: spr\_Rect[i].setPosition(370, 319); break;

case 11: spr\_Rect[i].setPosition(370, 264); break;

case 12: spr\_Rect[i].setPosition(370, 156); break;

case 13: spr\_Rect[i].setPosition(370, 99); break;

case 14: spr\_Rect[i].setPosition(480, 0); break;

case 15: spr\_Rect[i].setPosition(594, 0); break;

case 16: spr\_Rect[i].setPosition(648, 0); break;

case 17: spr\_Rect[i].setPosition(703, 0); break;

case 18: spr\_Rect[i].setPosition(760, 0); break;

case 19: spr\_Rect[i].setPosition(814, 0); break;

case 20: spr\_Rect[i].setPosition(870, 0); break;

case 21: spr\_Rect[i].setPosition(925, 0); break;

case 22: spr\_Rect[i].setPosition(1065, 99); break;

case 23: spr\_Rect[i].setPosition(1065, 156); break;

case 24: spr\_Rect[i].setPosition(1065, 264); break;

case 25: spr\_Rect[i].setPosition(1065, 319); break;

case 26: spr\_Rect[i].setPosition(1065, 431); break;

case 27: spr\_Rect[i].setPosition(1065, 545); break;

}

}

for (int i = 0; i < spr\_Token.size(); i++) spr\_Token[i].setTexture(tex\_Token[i]);

for (int i = 0; i < 6; i++) spr\_Dice[i].setTexture(tex\_Dice[i]);

for (int i = 0; i < 5; i++) { spr\_Button[i].setTexture(tex\_Button[i]); spr\_Button[i].setPosition(left, top += 60); }

ifs.open("Coordinat Map.txt", ios::in);

if (!ifs)

return;

ifs >> map;

ifs.close();

Menu menu(App);

menu.setImage();

control[0] = -1;

control[1] = -1;

j = 0;

menu.GetDataTokens(a, b, c, d, e, f);

for (int i = 0; i < 4; i++) { if (a[i] == 0) { control[j] = i; j++; } }

if (control[0] != -1) spr\_Token.erase(spr\_Token.begin() + control[0]);

if (control[1] != -1) spr\_Token.erase(spr\_Token.begin() + (control[1] - 1));

for (int i = 0, k = 0; i < 4; i++) {

if ((control[0] == -1 || control[0] != i) && (control[1] == -1 || control[1] != i)) {

spr\_Token[k].setTextureRect(IntRect(a[i], b[i], c[i], d[i]));

map.setStartPosition(g, h, k);

spr\_Token[k].setPosition(g[0], h[0]);

spr\_Token[k].setScale(e[i] / 3, f[i] / 3); k++;

}

else

continue;

}

Numplayer = 0;

player = new Player[spr\_Token.size()];

money = new Text[spr\_Token.size()];

for (int i = 0; i < spr\_Token.size(); i++) {

money[i].setFont(fnt\_Arial);

money[i].setCharacterSize(20);

money[i].setStyle(Text::Bold);

money[i].setFillColor(Color::Black);

}

for (int i = 0; i < spr\_Token.size(); i++) {

PlayerMoney = player[i].getMoneyPlayer();

money[i].setString(toStr(PlayerMoney) + '$');

switch (i) {

case 0: money[i].setPosition(30, 70); break;

case 1: money[i].setPosition(1280, 70); break;

case 2: money[i].setPosition(30, 722); break;

case 3: money[i].setPosition(1280, 722); break;

}

}

tex\_PlayerZone = new Texture[spr\_Token.size()];

tex\_TokenInPlayerZone = new Texture[spr\_Token.size()];

for (int i = 0; i < spr\_Token.size(); i++) {

tex\_PlayerZone[i].loadFromFile("images/color.png");

tex\_TokenInPlayerZone[i].loadFromFile("images/tokens.tga");

}

spr\_PlayerZone = new Sprite[spr\_Token.size()];

spr\_TokenInPlayerZone = new Sprite[spr\_Token.size()];

for (int i = 0; i < spr\_Token.size(); i++) {

spr\_PlayerZone[i].setTexture(tex\_PlayerZone[i]);

spr\_TokenInPlayerZone[i].setTexture(tex\_TokenInPlayerZone[i]);

switch (i) {

case 0:

spr\_PlayerZone[i].setTextureRect(IntRect(55, 104, 58, 58));

spr\_PlayerZone[i].setScale(Vector2f(2.f, 2.f));

spr\_PlayerZone[i].setPosition(0, 0);

for (int j = 0; j < 4; j++) {

if (a[j] == 0)

continue;

else {

spr\_TokenInPlayerZone[i].setTextureRect(IntRect(a[j], b[j], c[j], d[j]));

spr\_TokenInPlayerZone[i].setScale(Vector2f(e[j], f[j]));

a[j] = 0;

break;

}

}

spr\_TokenInPlayerZone[i].setPosition(30, 0);

break;

case 1:

spr\_PlayerZone[i].setTextureRect(IntRect(130, 104, 58, 58));

spr\_PlayerZone[i].setScale(Vector2f(2.f, 2.f));

spr\_PlayerZone[i].setPosition(1250, 0);

for (int j = 0; j < 4; j++) {

if (a[j] == 0)

continue;

else {

spr\_TokenInPlayerZone[i].setTextureRect(IntRect(a[j], b[j], c[j], d[j]));

spr\_TokenInPlayerZone[i].setScale(Vector2f(e[j], f[j]));

a[j] = 0;

break;

}

}

spr\_TokenInPlayerZone[i].setPosition(1280, 0);

break;

case 2:

spr\_PlayerZone[i].setTextureRect(IntRect(205, 104, 58, 58));

spr\_PlayerZone[i].setScale(Vector2f(2.f, 2.f));

spr\_PlayerZone[i].setPosition(0, 652);

for (int j = 0; j < 4; j++) {

if (a[j] == 0)

continue;

else {

spr\_TokenInPlayerZone[i].setTextureRect(IntRect(a[j], b[j], c[j], d[j]));

spr\_TokenInPlayerZone[i].setScale(Vector2f(e[j], f[j]));

a[j] = 0;

break;

}

}

spr\_TokenInPlayerZone[i].setPosition(30, 652);

break;

case 3:

spr\_PlayerZone[i].setTextureRect(IntRect(281, 104, 58, 58));

spr\_PlayerZone[i].setScale(Vector2f(2.f, 2.f));

spr\_PlayerZone[i].setPosition(1250, 652);

for (int j = 0; j < 4; j++) {

if (a[j] == 0)

continue;

else {

spr\_TokenInPlayerZone[i].setTextureRect(IntRect(a[j], b[j], c[j], d[j]));

spr\_TokenInPlayerZone[i].setScale(Vector2f(e[j], f[j]));

a[j] = 0;

break;

}

}

spr\_TokenInPlayerZone[i].setPosition(1280, 652);

break;

}

}

int Firm[] = { 1, 3, 5, 6, 8, 9, 11, 12, 13, 14, 15, 16, 18, 19, 21, 23, 24, 25, 26, 27, 28, 29, 31, 32, 34, 35, 37, 39 };

PlayerInJail = new int[spr\_Token.size()];

count = new int[spr\_Token.size()];

for (int i = 0; i < spr\_Token.size(); i++) count[i] = 0;

fl = false, isRoolDice = false, go = false, buy = false;

for (int i = 0; i < 4; i++) { a[i] = 0; b[i] = 0; }

for (int i = 0; i < spr\_Token.size(); i++) { count[i] = 0; PlayerInJail[i] = 0; }

Size = spr\_Token.size();

chance.setChance();

comm.setComm();

int numChance = 0, numComm = 0;

bool isChance = false, isComm = false;

bool nextPlayer = false;

while (App.isOpen())

{

Event event;

while (App.pollEvent(event))

{

if (event.type == Event::Closed)

App.close();

}

menuNum = 0;

posForBuy = player[Numplayer].getPosition();

spr\_Button[0].setColor(Color::White);

spr\_Button[1].setColor(Color::White);

spr\_Button[2].setColor(Color::White);

spr\_Button[3].setColor(Color::White);

if (Numplayer == spr\_Token.size())

Numplayer = 0;

App.clear();

App.draw(spr\_backgr);

App.draw(spr\_map);

if (posForBuy == 0 && player[Numplayer].getStart() == true) {

player[Numplayer].appMoney(200);

player[Numplayer].setStart(false);

}

if (IntRect(300, 60, 58, 58).contains(Mouse::getPosition(App))) {

spr\_Button[0].setColor(Color::Blue);

menuNum = 1;

}

if (IntRect(300, 120, 58, 58).contains(Mouse::getPosition(App))) {

spr\_Button[1].setColor(Color::Blue);

menuNum = 2;

}

if (IntRect(300, 180, 58, 58).contains(Mouse::getPosition(App))) {

spr\_Button[2].setColor(Color::Blue);

menuNum = 3;

}

if (IntRect(300, 240, 58, 58).contains(Mouse::getPosition(App))) {

spr\_Button[3].setColor(Color::Blue);

menuNum = 4;

}

if (Mouse::isButtonPressed(Mouse::Left)) {

if (menuNum == 1 && isRoolDice == false && go == false) {

player[Numplayer].ThrowDice(point);

fl = true;

isRoolDice = true;

player[Numplayer].setStart(true);

if (point[0] != point[1]) {

spr\_Dice[point[0] - 1].setPosition(600, 300);

spr\_Dice[point[1] - 1].setPosition(700, 300);

App.draw(spr\_Dice[point[0] - 1]);

App.draw(spr\_Dice[point[1] - 1]);

}

else {

spr\_Dice[point[0] - 1].setPosition(600, 300);

App.draw(spr\_Dice[point[0] - 1]);

spr\_Dice[point[0] - 1].setPosition(700, 300);

App.draw(spr\_Dice[point[1] - 1]);

}

if (PlayerInJail[Numplayer] == 0) {

pos = player[Numplayer].getPosition();

nextPosition = (pos + (point[0] + point[1]));

go = true;

}

else {

if (point[0] == point[1]) {

pos = 10;

nextPosition = (pos + (point[0] + point[1]));

PlayerInJail[Numplayer] = 0;

go = true;

}

else ++count[Numplayer];

if (count[Numplayer] == 3) {player[Numplayer].getOutOfJail();

PlayerInJail[Numplayer] = 0;

pos = 10;

nextPosition = (pos + (point[0] + point[1]));

go = true;

}

}

sleep(\*new Time(microseconds(100)));

}

if (menuNum == 2 && isRoolDice == true && go == false) {

if ((Numplayer + 1) >= spr\_Token.size())

Numplayer = 0;

else ++Numplayer;

pos = player[Numplayer].getPosition();

isRoolDice = false;

if (menuNum == 3 && go == false) {

int pos = player[Numplayer].getPosition();

for (int i = 0; i < 28; i++) if (pos == Firm[i]) { buy = true; break; }

if (map.isBuy(pos) == false && buy) {

player[Numplayer].buyFirm(pos, map);

int RectB;

for (int i = 0; i < 28; i++) {

if (pos == Firm[i])

RectB = i;

}

switch (Numplayer) {

case 0: spr\_Rect[RectB].setColor(Color::Blue); break;

case 1: spr\_Rect[RectB].setColor(Color::Green); break;

case 2: spr\_Rect[RectB].setColor(Color::Color(241, 156, 187)); break;

case 3: spr\_Rect[RectB].setColor(Color::Red); break;

}

buy = false;

}

}

if (menuNum == 4) {

menu.trade(App, player, map, Size, Numplayer, Firm, spr\_Token, money);

int size, Cast = 0;

for (int i = 0; i < spr\_Token.size(); i++) {

size = player[i].getSize();

int \*arr = new int[size];

player[i].setPositionMyFirm(arr);

for (int j = 0; j < size; j++) {

for (int k = 0; k < 28; k++) {

if (arr[j] == Firm[k]) {

arr[j] = k;

break;

}

}

switch (i) {

case 0: spr\_Rect[arr[j]].setColor(Color::Blue); break;

case 1: spr\_Rect[arr[j]].setColor(Color::Green); break;

case 2: spr\_Rect[arr[j]].setColor(Color::Color(241, 156, 187)); break;

case 3: spr\_Rect[arr[j]].setColor(Color::Red); break;

}

}

}

}

}

if (go == false) {

if ((pos == 2 || pos == 17 || pos == 33) && int mon = 0;

numComm = comm.getComm(mon);

player[Numplayer].appMoney(mon);

isComm = true;

player[Numplayer].setStart(false);

}

if (pos == 4 && player[Numplayer].getStart()) {

player[Numplayer].appMoney(-200);

player[Numplayer].setStart(false);

}

if ((pos == 7 || pos == 22 || pos == 36&& player[Numplayer].getStart())

int mon = 0, posChance = 0;

numChance = chance.getChance(posChance, mon);

if (mon == 0) {

nextPosition = posChance;

go = true; }

else if (posChance == 0) {

player[Numplayer].appMoney(mon);

}

isChance = true;

player[Numplayer].setStart(true); }

if (pos == 30) {

pos = player[Numplayer].getPosition();

map.goToJail(a, b, Numplayer);

player[Numplayer].setPosition(pos);

spr\_Token[Numplayer].setPosition(a[0], b[0]);

PlayerInJail[Numplayer] = 1;

}

if (pos == 38 && player[Numplayer].getStart()) {

player[Numplayer].appMoney(-100);

player[Numplayer].setStart(false);

}}

else if (player[Numplayer].getStart() && map.isBuy(pos) == true) {

player[Numplayer].Rent(player, pos, map, Numplayer, Size);

player[Numplayer].setStart(false);

}

if (fl == true) {

spr\_Dice[point[0] - 1].setPosition(600, 300);

App.draw(spr\_Dice[point[0] - 1]);

spr\_Dice[point[0] - 1].setPosition(700, 300);

App.draw(spr\_Dice[point[1] - 1]);

}

if (go == true) {

if (pos == nextPosition) go = false;

else {

map.setPosition(a, b, Numplayer, &pos, nextPosition);

player[Numplayer].setPosition(pos);

spr\_Token[Numplayer].setPosition(a[0], b[0]);

}

sleep(\*new Time(milliseconds(200)));

}

for (int i = 0; i < 5; i++) App.draw(spr\_Button[i]);

for (int i = 0; i < spr\_Token.size(); i++) App.draw(spr\_Token[i]);

for (int i = 0; i < spr\_Token.size(); i++) App.draw(spr\_PlayerZone[i]);

for (int i = 0; i < spr\_Token.size(); i++) App.draw(spr\_TokenInPlayerZone[i]);

for (int i = 0; i < spr\_Token.size(); i++) {

PlayerMoney = player[i].getMoneyPlayer();

money[i].setString(toStr(PlayerMoney) + '$');

App.draw(money[i]);

if (isChance) {

App.draw(spr\_Chance[numChance]);

if(Mouse::isButtonPressed(Mouse::Left)) isChance = false;

}

if (isComm) {

App.draw(spr\_Comm[numComm]);

if (Mouse::isButtonPressed(Mouse::Left)) isComm = false;

}}

for (int i = 0; i < 28; i++) App.draw(spr\_Rect[i]);

App.display();

sleep(\*new Time(milliseconds(100)));

}

}

string Mono ::toStr(int var)

{

stringstream tmp; tmp << var; return tmp.str();

}

File Player.h

#pragma once

#include<vector>

#include"Dice.h"

#include"Map.h"

#include<algorithm>

using namespace std;

class Player : public Dice {

private:

bool start;

long money;

int position;

int isJailCard;

vector<int> positionMyFirm;

vector<int> NumMyCorporation;

vector<int>::iterator it;

multimap<int, int> NumCorporation;

multimap<int, int>::iterator iNumCorporation;

public:

Player();

~Player();

void setMoney(int);

void setPosition(int);

int getPosition();

void buyFirm(int, Map&);

long getMoneyPlayer();

void appMoney(int);

void getOutOfJail();

bool getStart();

void setStart(bool);

int getSize();

void setPositionMyFirm(int \*);

void sortVector();

void Change(vector<int>, vector<int>, int \*, int, int);

};

File Player.cpp

#include "Player.h"

Player::Player() {

money = 2000;

position = 0;

start = false;

isJailCard = false;

}

Player::~Player() {

NumCorporation.insert(pair<int, int>(1, 1));

NumCorporation.insert(pair<int, int>(1, 3));

NumCorporation.insert(pair<int, int>(2, 6));

NumCorporation.insert(pair<int, int>(2, 8));

NumCorporation.insert(pair<int, int>(2, 9));

NumCorporation.insert(pair<int, int>(3, 11));

NumCorporation.insert(pair<int, int>(3, 13));

NumCorporation.insert(pair<int, int>(3, 14));

NumCorporation.insert(pair<int, int>(4, 16));

NumCorporation.insert(pair<int, int>(4, 18));

NumCorporation.insert(pair<int, int>(4, 19));

NumCorporation.insert(pair<int, int>(5, 21));

NumCorporation.insert(pair<int, int>(5, 23));

NumCorporation.insert(pair<int, int>(5, 24));

NumCorporation.insert(pair<int, int>(6, 26));

NumCorporation.insert(pair<int, int>(6, 27));

NumCorporation.insert(pair<int, int>(6, 29));

NumCorporation.insert(pair<int, int>(7, 31));

NumCorporation.insert(pair<int, int>(7, 32));

NumCorporation.insert(pair<int, int>(7, 34));

NumCorporation.insert(pair<int, int>(8, 37));

NumCorporation.insert(pair<int, int>(8, 39));

}

void Player::setMoney(int money) { this->money = money; }

void Player::setPosition(int position) { this->position = position; }

int Player::getPosition() { return position; }

void Player::buyFirm(int pos, Map &obj) {

obj.GetMap(pos, money);

positionMyFirm.push\_back(pos);

}

long Player::getMoneyPlayer() { return money; }

void Player::appMoney(int money) { this->money += money; }

void Player::getOutOfJail() { money -= 50; }

bool Player::getStart() { return start; }

void Player::setStart(bool start) { this->start = start; }

int Player::getSize() { return positionMyFirm.size(); }

void Player::setPositionMyFirm(int \*arr) {

for (int i = 0; i < positionMyFirm.size(); i++) {

arr[i] = positionMyFirm[i];

}

}

void Player::sortVector() { sort(positionMyFirm.begin(), positionMyFirm.end()); }

void Player::Change(vector<int> one, vector<int> two, int\* Firm, int money, int subtractmoney) {

vector<int>::iterator it;

this->money += money;

this->money -= subtractmoney;

int \*arr = new int[one.size()];

for (int i = 0; i < one.size(); i++) {

for (int j = 0; j < 28; j++) {

if (one[i] == j) {

arr[i] = Firm[j];

}

}

}

for (int i = 0; i < one.size(); i++) {

for (it = positionMyFirm.begin(); it != positionMyFirm.end(); it++) {

if (\*it == arr[i]) it = positionMyFirm.erase(it);

if (it == positionMyFirm.end()) break;

}

}

delete[]arr;

arr = new int[two.size()];

for (int i = 0; i < two.size(); i++) {

for (int j = 0; j < 28; j++) {

if (two[i] == j) {

arr[i] = Firm[j];

}

}

}

for (int i = 0; i < two.size(); i++) {

positionMyFirm.push\_back(arr[i]);

}

}

void Player::Rent(Player \*player, int pos, Map &map, int CurrentPlayer, int numPlayer) {

int pl = 4, mon = 0;

bool Corp;

for (int i = 0; i < numPlayer; i++) {

for (int j = 0; j < player[i].positionMyFirm.size(); j++) {

if (player[i].positionMyFirm[j] == pos) {

pl = i;

break;

}

}

if (pl != 4)

break;

}

if (pl != CurrentPlayer) {

Corp = player[pl].isCorporation(pos, pl);

if (Corp && pl != CurrentPlayer) {

mon = map.getInfo(pos, 1);

player[CurrentPlayer].money -= mon;

player[pl].money += mon;

}

else {

mon = map.getInfo(pos, 0);

player[CurrentPlayer].money -= mon;

player[pl].money += mon;

}

}

}

bool Player::isCorporation(int pos, int num) {

NumMyCorporation.clear();

int count = 0, m = 0, l = 0;

for (iNumCorporation = NumCorporation.begin(); iNumCorporation != NumCorporation.end(); iNumCorporation++) {

l = 0;

if (m != (\*iNumCorporation).first) m = 0;

for (int j = 0; j < positionMyFirm.size(); j++) {

if ((\*iNumCorporation).second == positionMyFirm[j] && m == 0) {

m = (\*iNumCorporation).first;

++count;

break;

}

else if ((\*iNumCorporation).second == positionMyFirm[j]) {

if (m == (\*iNumCorporation).first) {

++count;

if ((\*iNumCorporation).second == 3) {

NumMyCorporation.push\_back((\*iNumCorporation).first);

break;

}

else if ((\*iNumCorporation).second == 39) {

NumMyCorporation.push\_back((\*iNumCorporation).first);

break;

}

++iNumCorporation;

if ((\*iNumCorporation).first == m) {

for (int k = 0; k < positionMyFirm.size(); k++) {

if ((\*iNumCorporation).second == positionMyFirm[k]) {

NumMyCorporation.push\_back((\*iNumCorporation).first);

cout << "Corporation" << endl;

l = 1;

break;

}}

if (l == 1)

break;

}}

else m = 0;

}}}

for (iNumCorporation = NumCorporation.begin(); iNumCorporation != NumCorporation.end(); iNumCorporation++) {

for(it = NumMyCorporation.begin(); it != NumMyCorporation.end(); it++)

if ((\*iNumCorporation).first == \*it)

if((\*iNumCorporation).second == pos)

return true;

}

return false;

}

File Tokens.h

#pragma once

#include<iostream>

#include<map>

#include"Coordinate.h"

#include<fstream>

using namespace std;

class Tokens {

private:

map<Coordinate, bool> tokens;

map<Coordinate, bool>::iterator itokens;

public:

Tokens();

~Tokens();

friend ifstream& operator >>(ifstream &, Tokens &);

void getToken(int\*, int\*, int\*, int\*, double\*, double\*);

};

File Tokens.cpp

#include "Tokens.h"

Tokens::Tokens() { }

Tokens::~Tokens() { }

ifstream& operator >> (ifstream &ifs, Tokens &obj)

{

Coordinate temp;

obj.itokens = obj.tokens.begin();

for (int i = 0; i < 6; i++) {

ifs >> temp.x;

ifs >> temp.y;

ifs >> temp.Wight;

ifs >> temp.Height;

ifs >> temp.vec2f1;

ifs >> temp.vec2f2;

obj.tokens[temp] = false;

}

return ifs;

}

void Tokens::getToken(int\* a, int\* b, int\* c, int\* d, double\* e, double\* f) {

Coordinate temp;

temp.x = \*a;

temp.y = \*b;

temp.Wight = \*c;

temp.Height = \*d;

int i = 0;

for (itokens = tokens.begin(); itokens != tokens.end(); itokens++) {if ((\*itokens).first == temp && (\*itokens).second == true) {

i++;

tokens[temp] = false;

if ((++itokens) == tokens.end()) {

for (itokens = tokens.begin(); itokens != tokens.end(); itokens++) {if ((\*itokens).second == false) { (\*itokens).second = true;

\*a = (\*itokens).first.x;

\*b = (\*itokens).first.y;

\*c = (\*itokens).first.Wight;

\*d = (\*itokens).first.Height;

\*e = (\*itokens).first.vec2f1;

\*f = (\*itokens).first.vec2f2;

break;

}

}

break;

}

else {

int count = 0;

map<Coordinate, bool>::iterator it;

for (it = itokens; it != tokens.end(); it++) {

if ((\*it).second == false) { (\*it).second = true;

\*a = (\*it).first.x;

\*b = (\*it).first.y;

\*c = (\*it).first.Wight;

\*d = (\*it).first.Height;

\*e = (\*it).first.vec2f1;

\*f = (\*it).first.vec2f2;

count++;

break;

}

}

if (count == 0) {

for (it = tokens.begin(); it != itokens; it++) {

if ((\*it).second == false) {

(\*it).second = true;

\*a = (\*it).first.x;

\*b = (\*it).first.y;

\*c = (\*it).first.Wight;

\*d = (\*it).first.Height;

\*e = (\*it).first.vec2f1;

\*f = (\*it).first.vec2f2;

break;

}

}

break;

}

}

break;

}

}

if (i == 0) {

bool flag = false;

itokens = tokens.begin();

while (!flag && itokens != tokens.end()) {

if ((\*itokens).second == false) {

(\*itokens).second = true;

\*a = (\*itokens).first.x;

\*b = (\*itokens).first.y;

\*c = (\*itokens).first.Wight;

\*d = (\*itokens).first.Height;

\*e = (\*itokens).first.vec2f1;

\*f = (\*itokens).first.vec2f2;

flag = true;

}

else

itokens++;

}

}

}

File Source.cpp

#include"Mono.h"

int main() {

Mono monopoly;

monopoly.start();

return 0;

}

# ДОДАТОК B

Електронна версія курсової роботи