SLOVAK UNIVERSITY OF TECHNOLOGY IN BRATISLAVA

FACULTY OF INFORMATICS AND INFORMATION TECHNOLOGY

**Checkpoint 1**

Siegecraft

**Martin Szabo**

2022

Table of Contents

Home 5

1 Description of the basic objectives and operation of the game 6

2 Game control design 7

3 Requirements 8

3.1 Collisions 8

3.2 Difficulty levels 8

3.3 Winning condition 8

3.4 Animation 9

4 Design of all game screens and menus 10

4.1 Main menu 10

4.2 Settings 10

4.3 Control 11

4.4 Game Over 11

5 OOP design 12

5.1 class Vector 12

5.2 class GameObject 12

5.3 class Unit extends GameObject 13

5.4 class Character extends Unit 13

5.5 class Peasant extends Character 14

5.6 class Guard extends Character 14

5.7 class Mage extends Character 14

5.8 class Building extends Unit 14

Home

This semester project from the course Fundamentals of Interactive Application Development will be a game created in JavaScript.

This game will be an RTS. In it you must build bases and fight or repel AI attacks. The game will have several types of units and several buildings which will have unique features. Also, the game will have 3 levels of difficulty.

1. Description of the basic objectives and operation of the game

In an RTS, the player places structures and maneuvers multiple units under his control to secure areas of the map and/or destroy the opponent's assets. In a typical RTS game, additional units and structures can be created, usually limited by the requirement to expend accumulated resources. These resources, in turn, are obtained by controlling special points on the map and/or owning certain types of units and structures designed for this purpose. More specifically, a typical game in the RTS genre involves resource accumulation, base building, in-game technological development, and unit control.

The screen is divided into a map area that shows the game world and terrain, units and buildings, and an interface overlay that contains command and production controls. The player is usually provided with an isometric view of the world or a direct overhead view. Players move around the screen and issue commands primarily with the mouse.

The game consists of the player being placed somewhere on the map with a few units or a building that can build more units/buildings. The player must build specific buildings to unlock more advanced units in the technology tree. The player is required to build an army and use them to either defend against a form of attack by an enemy AI that also has an army, or to attack and eliminate enemies that own bases with their own unit production capabilities.

1. Game control design

The controls of the RTS game are quite simple. Left click on the AI to give commands to units. Move the mouse to select where you want to execute the command. We can also use keyboard shortcuts for these commands, and then use the mouse to select only the place where the unit should execute the command.

|  |  |
| --- | --- |
| Button | Action |
| Left mouse button | Interaction |
| ESC | Pause |
| A | Attack |
| B | Build |

1. Requirements
   1. Collisions

In this game collisions will be counted as a rectangle around the object and when one rectangle starts to collide with the other, they can interact with each other.

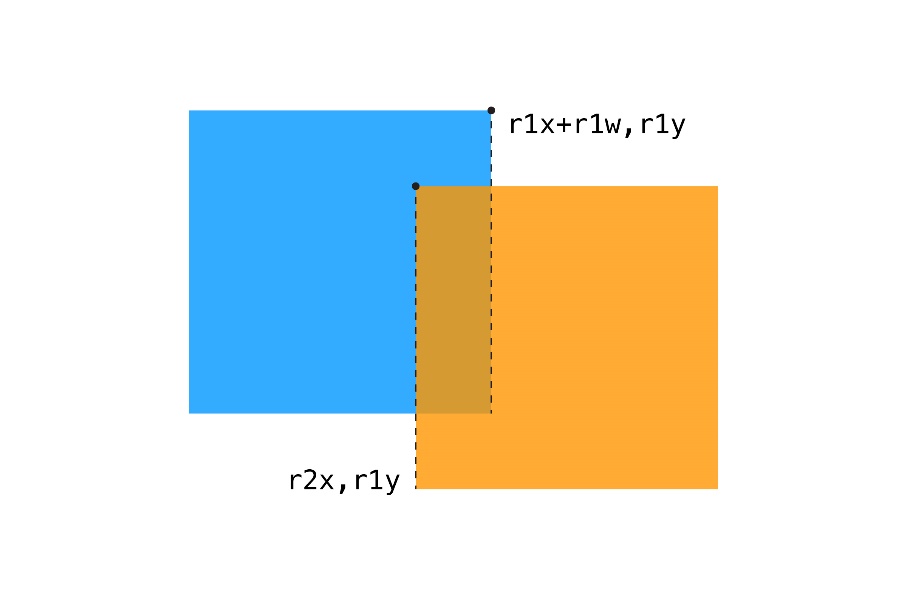


Image 1 Collision of two rectangles

Objects will interfere if all of the following conditions apply

r1.X < r2.X + r2.width

r1.X + r1.width > r2.X

r1.Y < r2.Y + r2.height

r1.Y + r1.height > r2.Y

I can extend this concept of collision with an offset so that objects do not have to be over each other to interact with each other.

* 1. Difficulty levels

The game will contain three levels of difficulty. Easy, medium, and hard. These difficulty levels will add more damage and lives to the enemy. But also, will be able to enlarge your army faster.

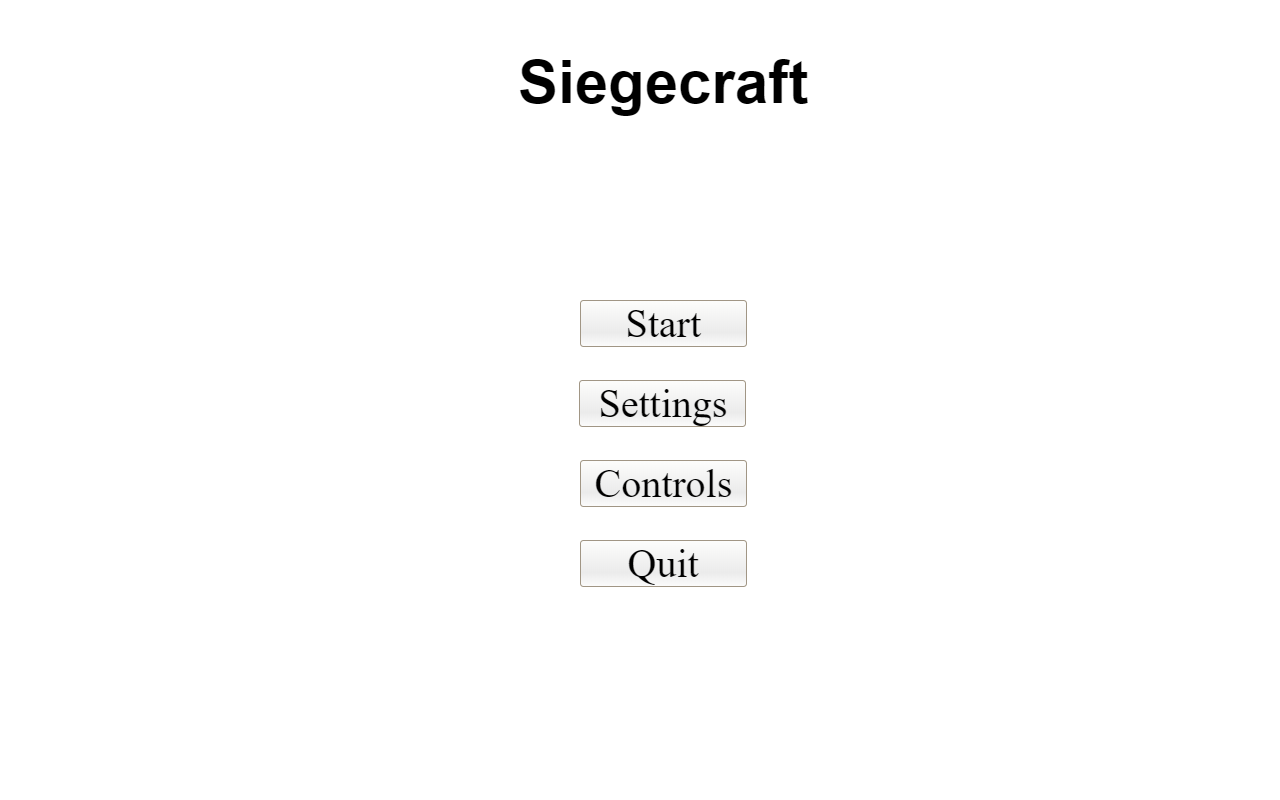
* 1. Winning condition

The player manages to win the game when the enemy has no buildings or units left on the map. That is, eliminating the enemy. But for that the player will have to think strategic.

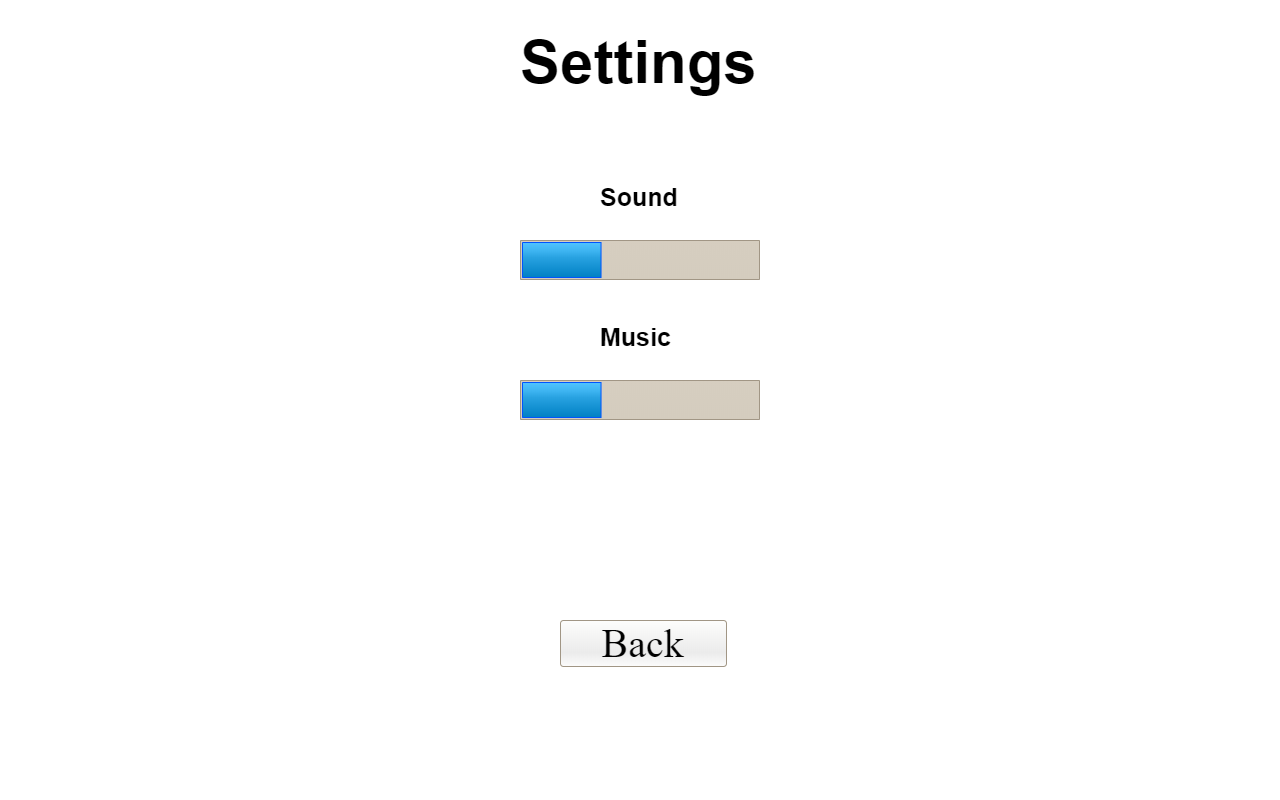
* 1. Animation

Animations will be performed by continuously changing the position of the object on the map. But the textures will also change. So called Sprite Animations. Characters will have their texture change when they run, attack or are low on health.

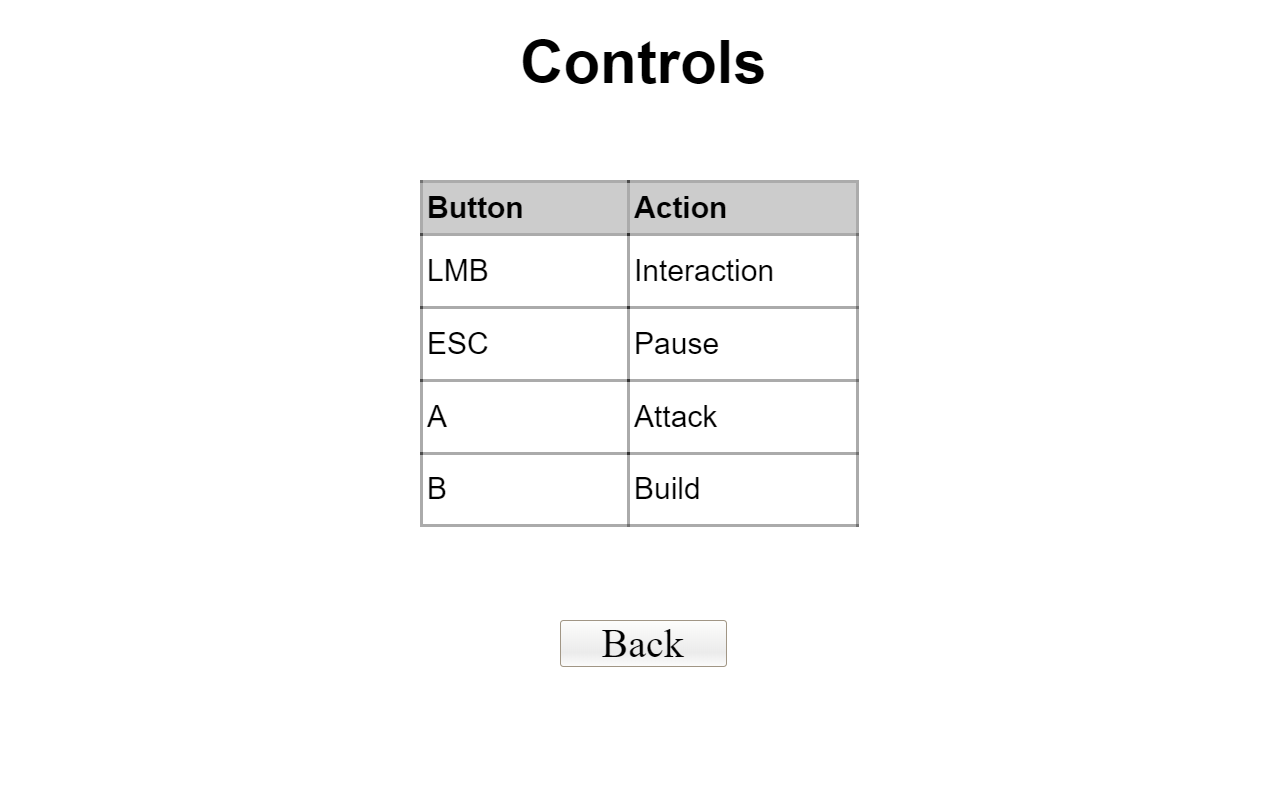
1. Design of all game screens and menus
   1. Main menu



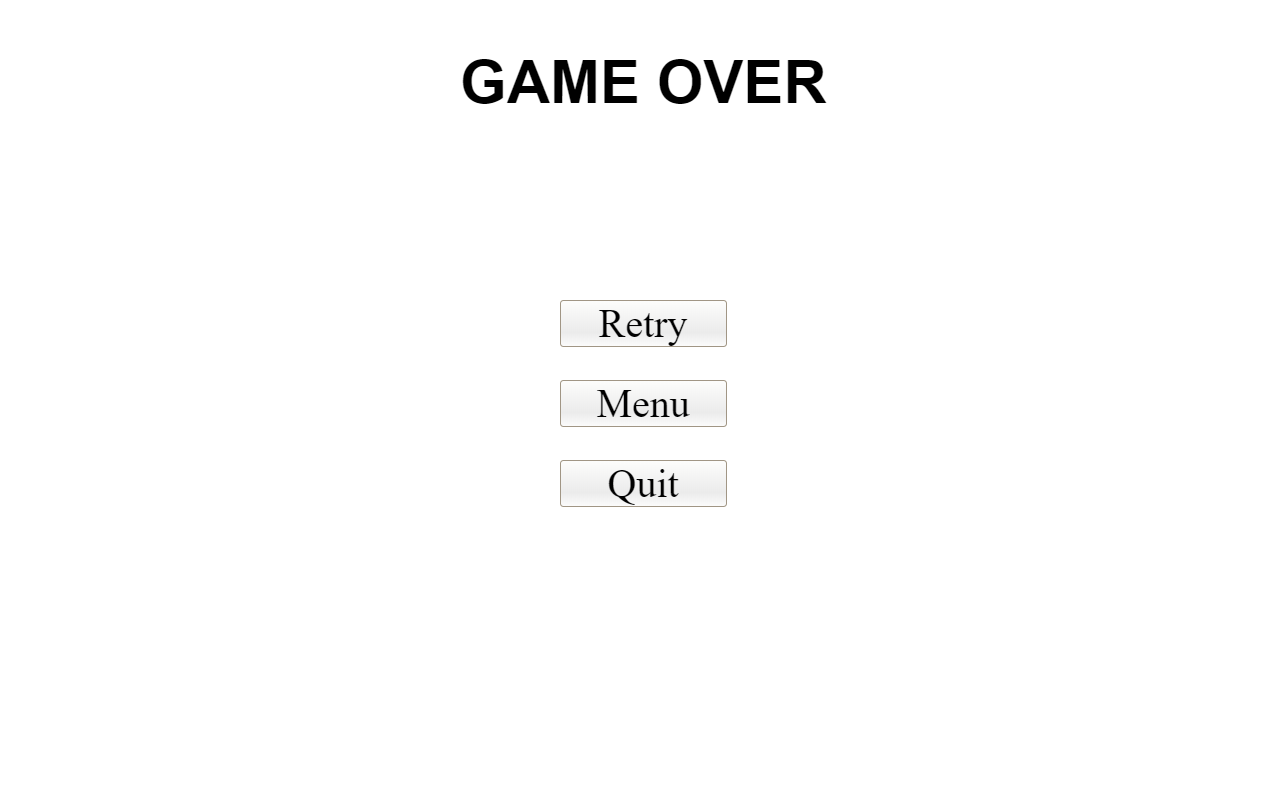
* 1. Settings



* 1. Control



* 1. Game Over



1. OOP design
   1. class Vector

A simple object consisting of X and Y attributes

|  |  |  |
| --- | --- | --- |
| Attribute | Type | Description |
| X | Number | X component of the vector |
| Y | Number | Y component of the vector |

* 1. class GameObject

The basics of an object that we can place in a game with certain position, dimensions, and textures.

|  |  |  |
| --- | --- | --- |
| Attribute | Type | Description |
| Texture | String | Path to the texture that will represent the object |
| Position | Object | Vector object. Represents where to place the object |
| Dimensions | Object | Vector object. Represents how big the object will be |

|  |  |
| --- | --- |
| Behaviour | Description |
| drawTexture(texture) | Render a texture to represent the object at the position specified by the position attribute. |
| update(position, velocity) | Moves the object to the position at the given speed. |
| isColliding(colidingObject, offset) | Returns a boolean if the objects collide. It also allows to set the offset if the objects should collide rather than touching directly. |

* 1. class Unit extends GameObject

This object has the basic properties that every object that the player or enemy will be able to place in the game should have. It inherits attributes and behaviors directly from the GameObject and extends them with three new attributes object type, armor (armor) and HP (lives).

|  |  |  |
| --- | --- | --- |
| Attribute | Type | Description |
| Type | String | Unit type |
| HP | Number | Number of unit lives |
| Armor | Number | Unit resistance to attacks |

* 1. class Character extends Unit

This object will augment the Unit object with human behavior. Attack, Talk. It will be all live.

|  |  |
| --- | --- |
| Behavior | Description |
| attack(unit, range) | Attacks a unit from a certain distance and takes its lives |
| talk() | The unit will say the message |

* 1. class Peasant extends Character

The Peasant will be used to occupy the conquered base and to extract resources.

|  |  |
| --- | --- |
| Behavior | Description |
| build() | Occupies a conquered or neutral stronghold and his faction's stronghold stands there |
| mine() | Begins to mine raw materials for forts |

* 1. class Guard extends Character

A Soldier is a unit that is resistant to melee attacks.

* 1. class Mage extends Character

The Mage is a ranged unit with more attack at the cost of less armor and lives.

* 1. class Building extends Unit

This object will extend the Unit object with the properties that all inanimate units that players can create should have. The Unit object will extend the behavior of the created unit.

|  |  |
| --- | --- |
| Behavior | Description |
| createUnit() | Creates a unit corresponding to the building type. |