Computer Graphics Project

AEI Tower

Authors:	Malika Uskembayeva
	Adam Gembala
	Robert Lotawiec
Field of study:	Informatics sem. VI
Academic year:	2021/2022

1. Description of the project

AEI Tower is a Icy Tower remake inspired by the day-to-day reality of students of the Faculty of Automatic Control, Electronics And Computer Science. The game consists in controlling a character representing a student of the AEI faculty as he/she makes his/her way to the top of the Mage Tower by jumping on successive stairs and advancing to successive floors of the faculty. The character is controlled with the keyboard. It is a platform game set in a tower, where the player's goal is to jump from one "floor" to the next and go as high as possible without falling and plunging off the screen.

2. Task analysis

At the beginning of semester project assumed following features:

- Gameplay divided into levels – each with it's own background and platform texture.
- Randomly generated challenges for player.
- Animations of the player and surroundings.
- Generating platforms should be performed while player progresses through the game.
- Score system – player should collect ECTS points.

In the final version we managed to implement almost all functionalities.

Player levels up after collecting 10 ECTS on each floor. Game form time to time spawns mystery chest for the player, which contains random reward or surprise. Animation of the player has two states – idle and jumping. There is constant amount of the platforms in the game, when platform leaves camera view it is destroyed and new platform is generated. Initial concept of score system was extended by player lifes – during game player collects not only ECTS points but also materials from previous year, that counted as player lifes.

3. External specification

Instruction: Link

After launching program game displays main menu.

When player clicks start button game begins on the base platform.

Character can be controlled using following keys:

- W – jump
- A – move left
- D – move right

Description of UI:

In order to complete the game player has to jump on platforms. Each platform is worth 1 ECTS. Player can collect ECTS as he first time lands on platform or when platform is destroyed after falling of the screen.

As it was mentioned before player can also collect materials from the previous year.

It is also possible to collect mystery chests:

After opening chest player can encounter random events:

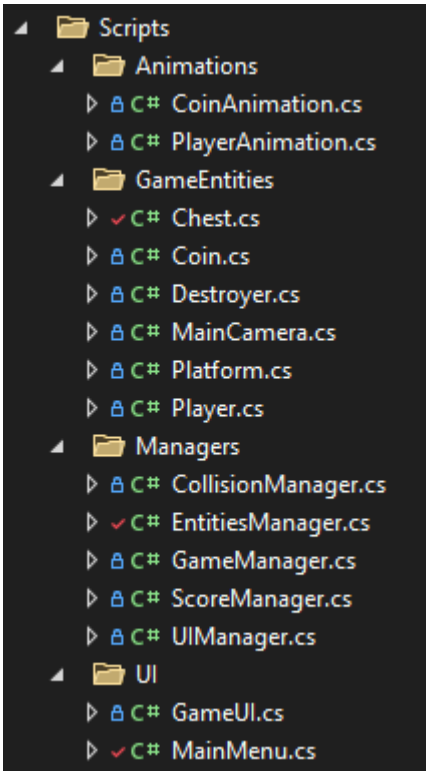
- Finding materials on the internet – Adds one material.
- Professor changed questions – Subtracts one material.
- Lucky exam – Adds 5 ECTS points.
- Empty chest – Does nothing.

When player wins/looses one of the screens is displayed:

4. Internal Specification

Source code: [GitHub Repository]

Scripts structure

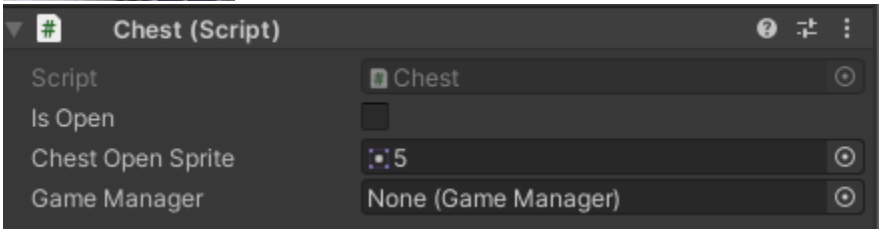


Scripts are divided into categories:

- Animations
  Scripts that animate game entities.
- Game entities
  Logic of game entities such as movement and collisions.
- Managers
  Controllers of certain parts of the game.
- UI
  User interface logic.

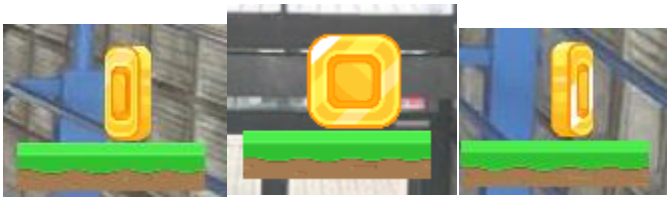
Game entities

- Chest



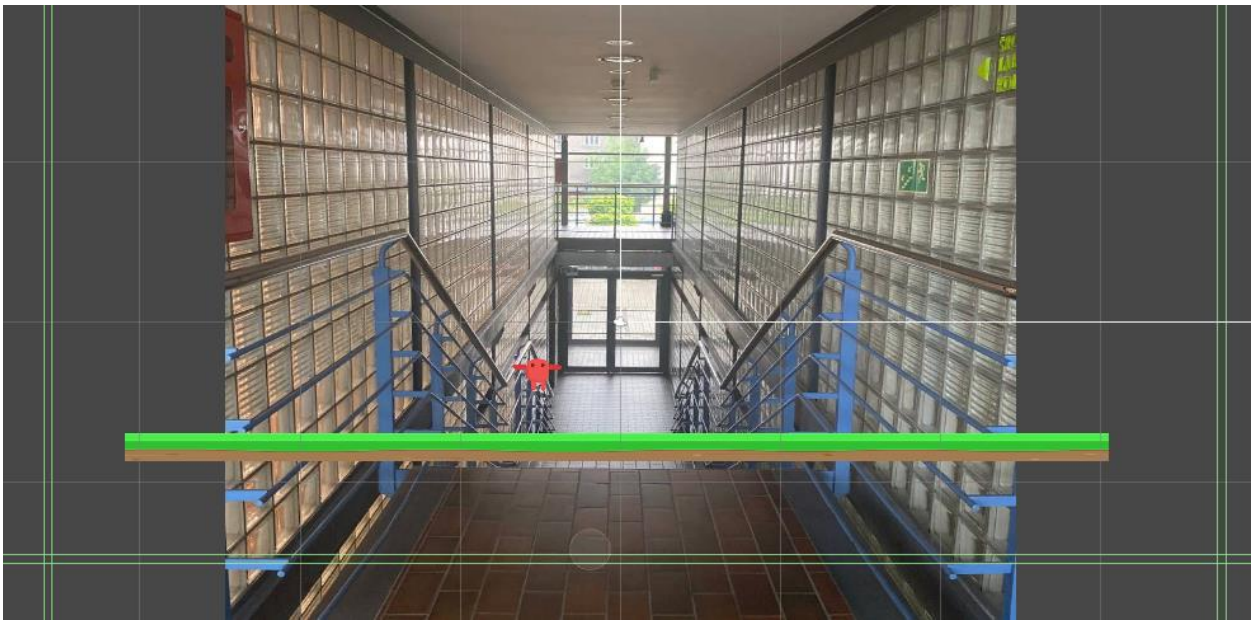
Chest can have two states – opened and closed. When spawned, chest is by default closed, when player collides with chest it opens and randomly generates event as it was described in external specification.

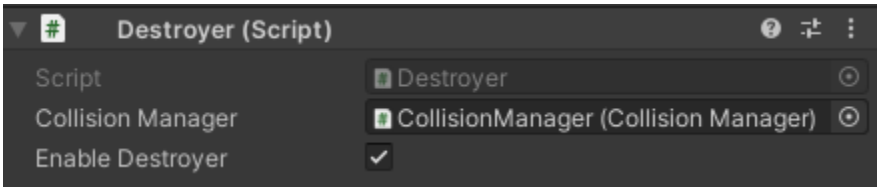
- Coin



Coin represents materials from previous year, which are counted as player lifes. When player collides with coin it calls score manager to add number of points assigned to it and calls entities manager to remove it from the game.

- Destroyer

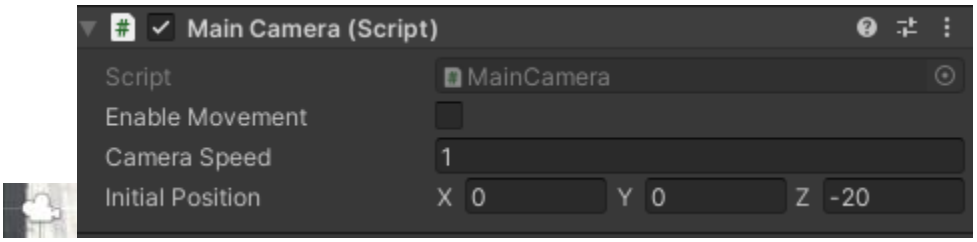




Destroyer is an empty object, which consists of box collider. It's job is to detect objects that are out of the range of the game area, and as it name suggests destroy them. Destroyer depending on colliding object type can before destroying object take some specific action:

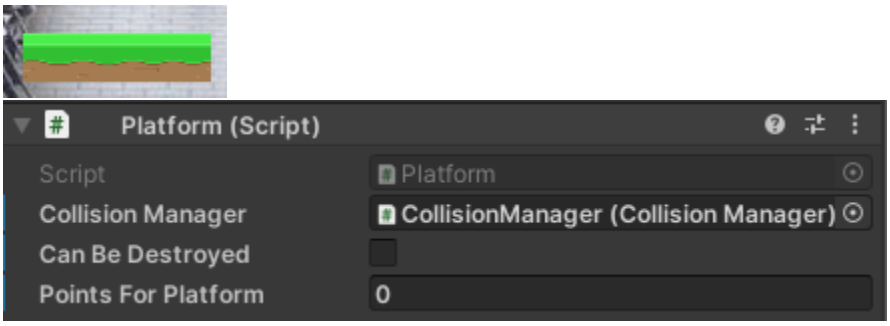
  - Player – calls score manager to decrease player's lifes (which sometimes end the game).
  - Platform – if platform has still assigned points to it, destroyer calls score manager to add this points to overall score and then destroys that platform.
  - Coin – Just destroys it.
  - Chest – Just destroys it.

- Main camera



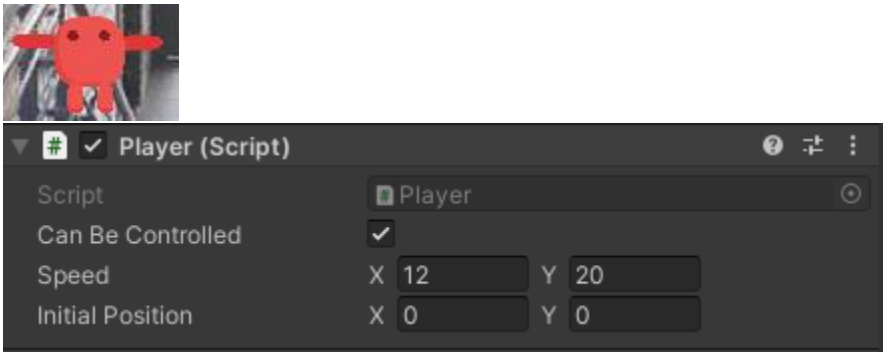
Main camera moves the game view up with certain speed.

- Platform



Platform is an object that detects only collisions that come from above. When player steps on platform it calls score manager to add ECTS point assigned to it.

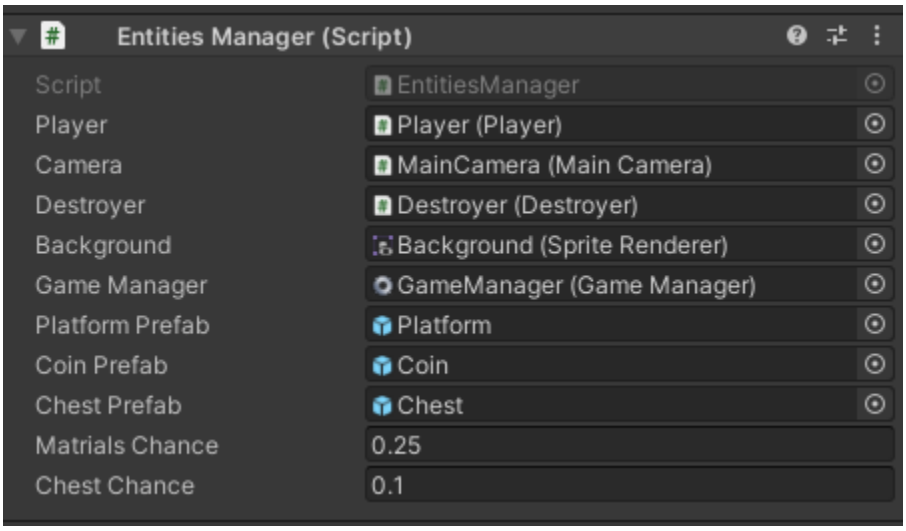
- Player



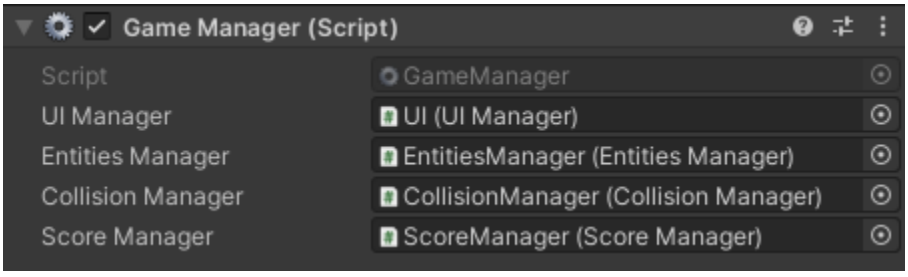
Player is a main character in the game. It can be controlled in order to progress through the game.

Managers

- Collision manager
  Contains logic of collisions between various object types.
- Entities manager

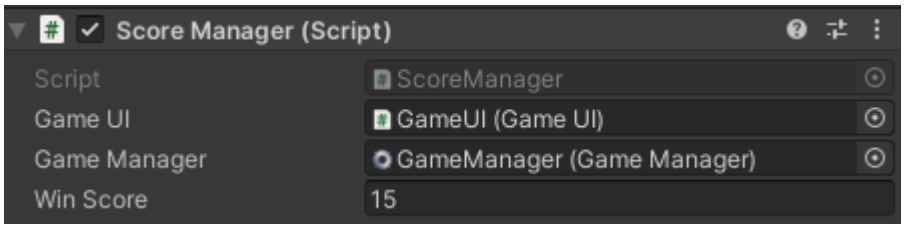


  Contains references to various types of game entities and prefabs. Controls lifetime of every object generated during game – can spawn and destroy platforms, chests and coins.
- Game manager



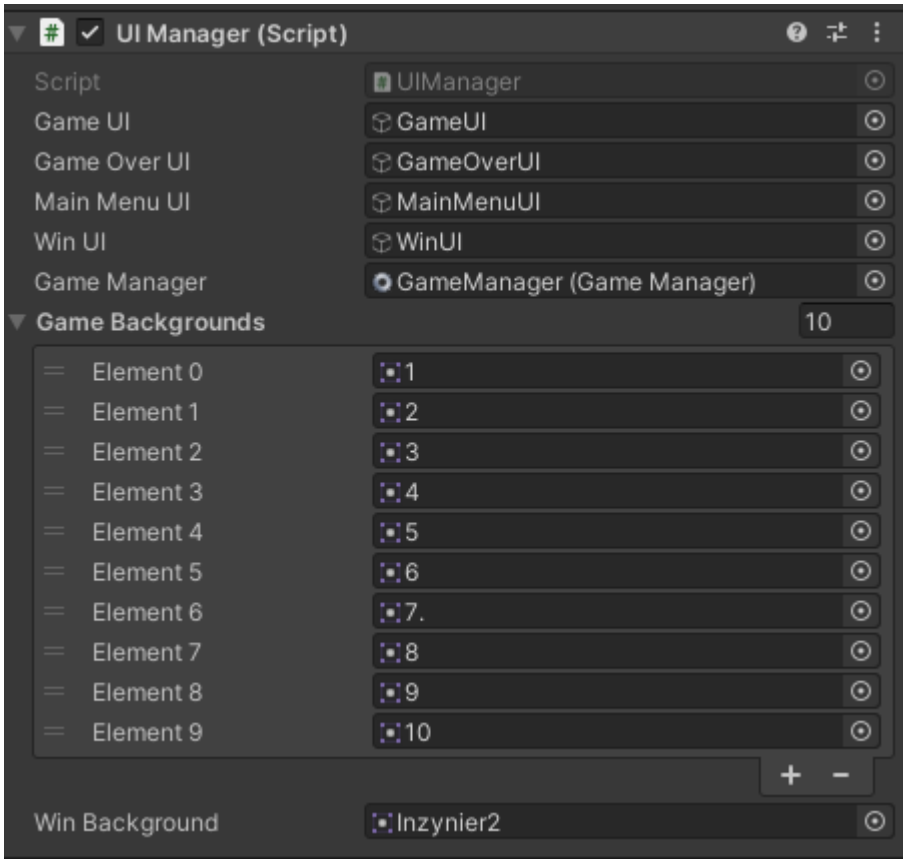
  Contains logic of events such as new game, game over or win. Game manager has references to all other managers, therefore can reset the state of the game.

- Score manager



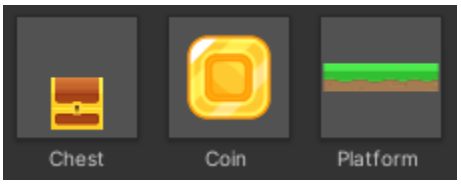
  Controls player score and lifes. Score manager can also decide if player completed the game.

- UI manager



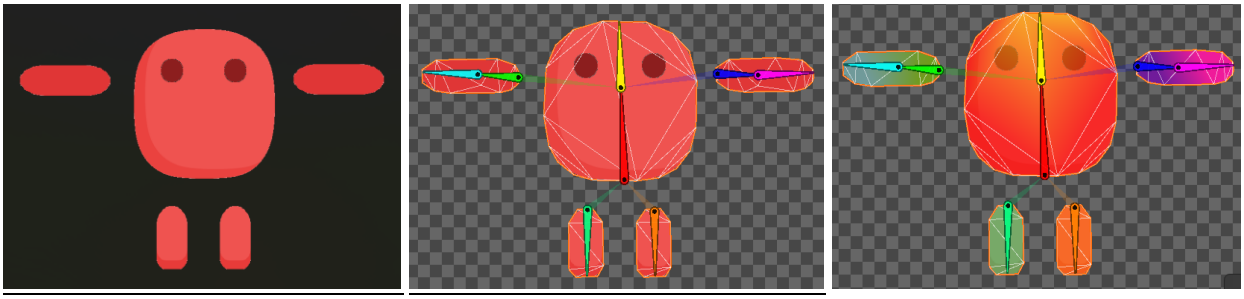
  UI manager changes currently active UI depending on game state. It can also control background as player progresses through the game and sets the win background when score manager reports that game was completed. Manager has a queue of messages to be displayed on the screen.

Prefabs



Entities that are generated during game are using prefabs as a templates for new objects.

Player sprite



Player character moves using bone animation. Character sprite was created from free assets available at Unity Hub[1].

[1] https://assetstore.unity.com/packages/2d/environments/free-platform-game-assets-85838

5. Testing and running

Game speed bug

During the early development of our game (before beta version) we observed a strange behavior of camera speed depending on the computer game was run on. The mystery was that the problem wasn't in performance of our code – other elements of the game were smooth on every machine. The most insane case was that when one of our team members launched the game on his laptop without power plugged in, the camera was barely moving, but after plugging it to the wall socket camera speed was enormous. Bug was hard to debug, but we managed to solve the problem by multiplying the camera speed by Time.deltaTime in MainCamera update method. We learned by experience that period of calling update method for game object depends on machine and that some laptops use their GPU only when they are plugged into wall socket.

6. Conclusions

In the final version of the project our team managed to implement almost all functionalities that were assumed at the beginning of the semester. Our team at the beginning of the semester decided to use Jira to track the progress of the project, but because development speed was vastly influenced by the fact that our team was learning Unity engine as the project grew, our team decided in first two weeks that we will resign from this idea. In the end we've learned a lot from development of this project, not only about Unity itself, but also about preparation of the assets and the design of the game.