Hotel Board Game

Graphics Unity Project 2021

Deadline: Sunday 20/06/2021

Contents

1 Introduction

The task is to recreate the board game "Hotel" in **Unity** game engine.

Hotel board game is a two-player board game, played in turns.

Hotel board game is defined by a fine set of rules. You may let your inner creativity wild and create another dimension of the board game, keeping its rules.

A fully completed game should consist of the following sections and components:

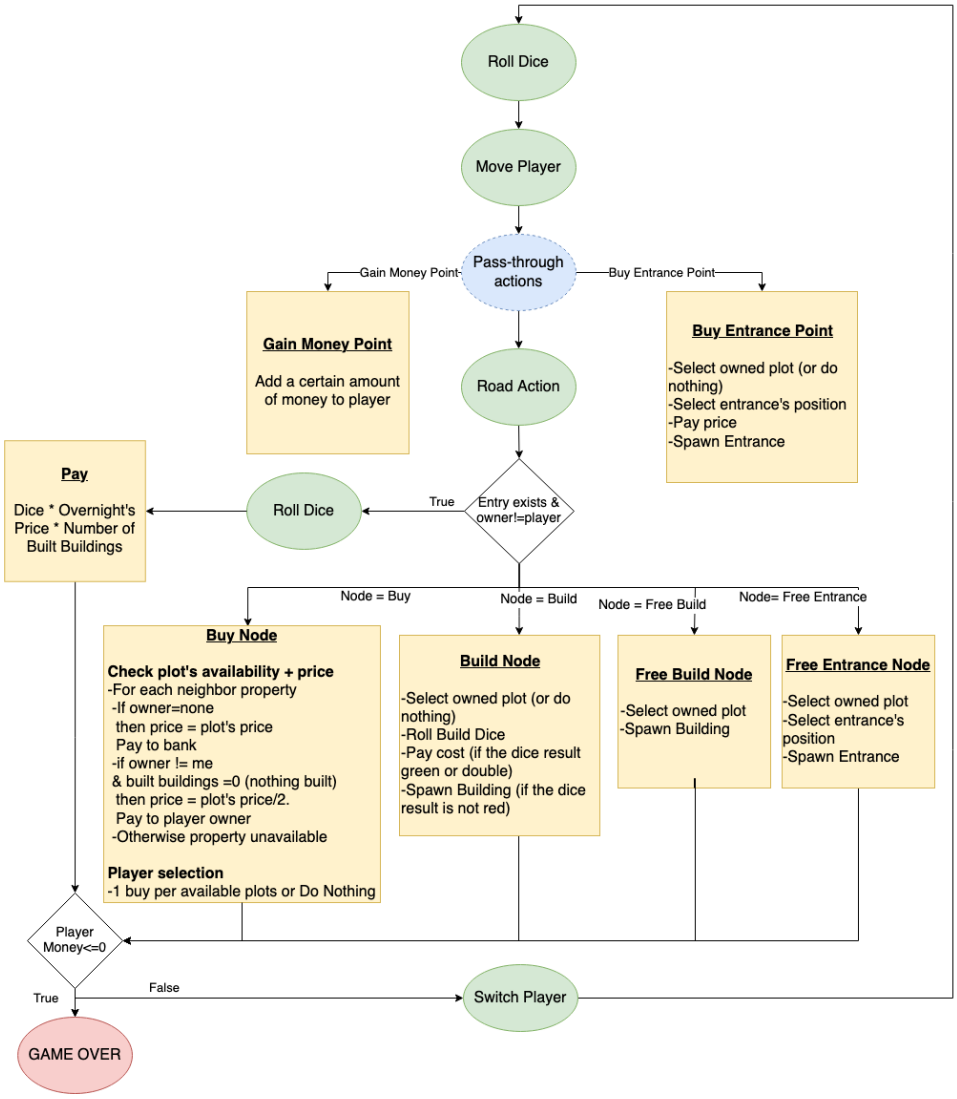
1. **Graphics**: 3D scene design, 3D objects, textures, materials, lights.
2. **Physics**: colliders, rigidbodies, forces .
3. **Scripting**: game logic and mechanics, event handling.
4. **Audio**: sound effects, music.
5. **UI**: text display, buttons

2 Hotel Board Game Overview

The concept of the Graphics Course's project for 2021, is associated with a very specific genre of games, the board games. Most popular board games already have published digital versions.

The board game that was chosen as the project's object is the Hotel Tycoon (created by Denys Fisher), due to both its simple and complex rules that it has. The Hotel is a multiplayer game from the category of fast-dealing property trading board games.

What is the Hotel and how is it played?

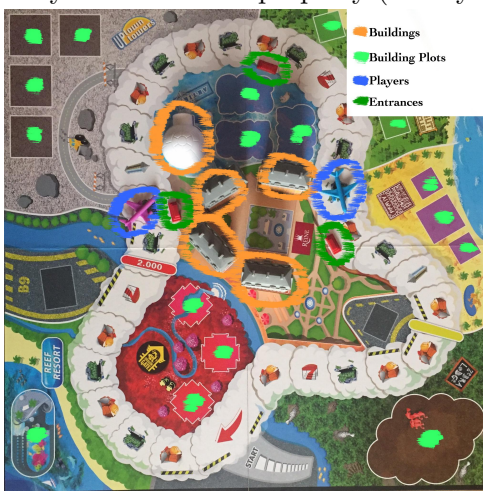


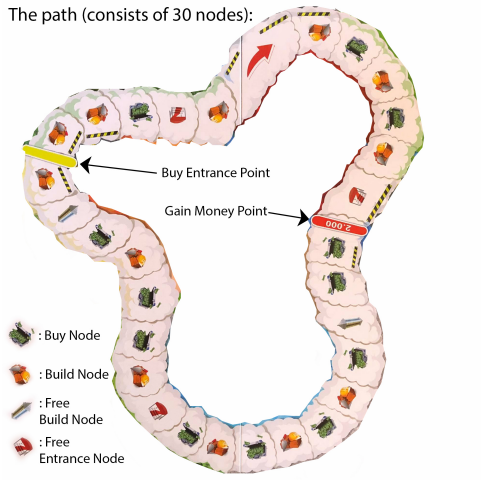
The Hotel's board consists of 2 to 4 players with money, plots, buildings/hotels, two hexahedral D6 dices (a regular hexahedral used for movement and payment and a specialized hexahedral dice used for building permits) and a path with nodes, on which the players move.

The game begins:

The game is played in rounds and the movement of each player in each round is determined by rolling a normal hexahedral D6 dice.

The hotel can played by 1 to 2 players, and their purpose is to buy plots, build buildings/hotels on them and put entrances on the nodes of the path, so their opponents may land on these nodes and pay for overnights until they have no other property (money and hotels) and lose.





More specifically, there are **plots** on which **buildings/hotels** can be built. In order to acquire a plot, a player should land on a **Buy Node** [2.2] (its functionality is described below) where he/she has the opportunity to buy if he/she wants one of his adjacent plots.

If players have bought some plots, they should build buildings/hotels on

them. In order to do this, players should land into specific nodes that determine whether the player is allowed to build or not. In some cases this requires the rolling of the specialized D6 **Build Dice** [2.3] in order to build (whose functionality is described below).

If players have built some buildings/hotels on their plots, they should place **entrances** on the nodes that are right next to their plots on which at least one building is built, with the purpose that their opponents land on these nodes and **pay** [2.5] for overnights (the payment process is described below). Players can obtain entrances for their hotels by two ways. The first way is players fall on the **Free Entrance Nodes** [2.2] and the second way is players pass through the **Buy Entrance Point** [2.4] (their functionalities are described below).

When players pass through the **Gain Money Point** [2.4], they receive a free amount of money (its functionality is described below).

If a player has built all the buildings on a plot, he/she can pay to add the expansion to that plot in order to increase its value.

A player should make other players go **bankrupt** [2.6], in order to win.

2.1 Turn and Move

The game is played in rounds and the steps of each player in each round are determined by rolling a normal hexahedral D6 dice. If the player rolls the dice and the result is 6 he/she can play again if he/she wants as soon as he/she completes the round. Also if a player lands on a node where another player has already landed, he/she automatically moves on the next available node.

2.2 Nodes

- **Buy Nodes**

  Players can buy a plot that is right next to one of the sides of the Buy Node on which they have landed (Players can only make one purchase each time on their turn).

  If players decide to buy it, they pay the price of the plot to the bank and they become the owners. In this case, they can build buildings/hotels on this plot (from the next time their turn comes again).

If one of the adjacent plots of the node, that they are, belongs to another player, but the other player has not built any building/hotel on it, they can buy it from this player (and this player can not deny). They just pay to the player the half of real price of this plot and they become the owners. This other player is obliged to sell it to them.

- **Build Nodes**

  If players have bought a plot of land, they must start building buildings on it and then add the extension. For this to happen, players must either land on a **Build Node** or a **Free Building Node**.

  In the first case, when players land into a build node, they can choose (if they want) only one plot that they own, on which not all of its buildings/hotels are already built, and declare how many buildings they want to build on this plot. It does not matter if the build node on which they have landed in is not next to the plot on which they want to build.

  Then, to get permission to build the declared buildings, the player must roll a special hexahedral dice named **Build Dice** (which will be explained below).

- **Free Build Nodes**

  In the second case, when players land on a Free Build Node, they can build for free a building/hotel on one of the plots that belong to them and they have not built all the buildings on the certain plot that they own. If the player has built all the buildings on a plot, then he/she can place the extension of this plot for free.

- **Free Entrance Nodes**

  When players land on a Free Entrance Node, they have the opportunity to choose one node (which is next to one of their plots with at least one building built) and place one free entrance on it, but only if no other entrance has already been placed in the specific node belonging to the same or another adjacent plot. This entrance is provided free of charge regardless of the entrances that players may have bought from the Buy Base Point.

2.3 Build Dice

When players declare on which plot and how many hotels wish to build, they should roll the Build dice to get permission to build those buildings.

The possible results of the dice are described below:

- **Red - 1 sides**

  The request for permission is rejected. Players have to wait until they land again in another Build Node to try again.

- **Green - 3 sides**

  The request for permission is accepted. Players pay the sum of buildings'/hotels' prices (which they wish to build) to the bank and then the buildings/hotels are built. Players are obliged to build if they are given permission, even if their money is not enough.

  If they have enough money, they pay the cost and build the buildings/hotels. If they do not have enough money, they have to sell their plots to other players with everything they contain (buildings/hotels and entrances). If they still do not have enough money or do not own any plot, then they go bankrupt and lose.

- **H - 1 side**

  The request for permission is accepted. Players build the buildings/hotels of the plot that they declared for free.

- **D - 1 side**

  The request for permission is accepted. Players pay double the sum of buildings'/hotels' prices (which they wish to build) to the bank and then the buildings/hotels are built. Players are obliged to build if they are given permission, even if their money is not enough.

  If they have enough money, they pay the cost and build the buildings/hotels. If they do not have enough money, they have to sell their plots to other players with everything they contain (buildings/hotels and entrances). If they still do not have enough money or do not own any plot, then they go bankrupt and lose.

2.4 Pass-Through Points

- **Gain Money Point**

  Every time players pass the Gain Money Point, they receive a certain amount (2000) from the bank.

- **Buy Entrance Point**

  Every time players pass the Buy Entrance Point, they can buy (only if they want) one entrance for each plot on which they have already built at least one building.

  Players pay the sum of entrances' values that they want to place (1 entrance for every 1 built plot) and obtain the entrances. Then, players select the nodes next to each of the hotels for which they bought the entrances and place the entrances to them.

  In no case, can the players buy entrances and place more than 1 on a certain plot.

2.5 Pay

When players land at a node with a hotel's entrance belonging to an opposing player, they should roll the regular D6 dice to determine how many overnights they will spend at this hotel (from 1 to 6 nights). Depending on the number of buildings that are built on the plot to which the entrance belongs, the overnight's price of this hotel and the result of the dice (the number of overnights), they will pay the bill at the opponent's hotel. The bill equals the number of buildings/hotels, which are built in this certain plot, multiplied with the result of the D6 dice and the overnight's price in this hotel.

2.6 Bankruptcy / The Winner

When players do not have enough money to pay for overnights at a hotel or build a building while the building permission was approved, they go into bankruptcy and lose. The last player with hotels and money is declared the winner.

3 Requirements

3.1 Minimum Requirements (50%)

1. **Graphics (10%)**: One (or more) Unity 3D scene with the necessary 3D objects. All objects must have different materials with at least one basic texture on them.

   You should create a scene with a board that consists of a path of nodes (at least ten), plots (at least three), hotels (at least two on each plot), pass-through points (at least one gain money point), and one player. Extensions of hotels are not required.

   You should use Unity's 3D models (cubes, spheres, ..) to create your scene by transforming them to give them the shape you wish.

2. **Physics (5%)**: Objects must have colliders and rigidbodies (if needed), so that they behave naturally and as realistic as possible.

3. **Game logic and mechanics (20%)**: At the minimum requirements, your project must be able to provide the following functionality:

   Singleplayer Mode: One player **moves [2.1]** on a path that consists of two types of nodes **(Buy and Build [2.2])** and one type of pass-through point **(Gain Money) [2.4]**. The player must be able to move on the path, buy plots, build buildings on them and get an amount from the bank when passing the Gain Money Point. You should create a script that produces random results instead of the two dices (one for **movement [2.1]** and one for **build [2.3]**).

Note 1: Regarding the functionality of the **Build Node [2.2]**, your game logic does not need to include the player's ability to declare how many buildings the player wants to build, provided that the player has the ability to build only one building at a time.

Note 2: Your game logic does not need to include the player's ability to add an extension to his/her hotel.

4. **UI & Cameras (10%)**: The UI components need to be handled by scripts so they are updated in real-time.
   - At least one UI text object presenting useful information about the player such as **money**, **owned buildings**, the result of dices etc.
   - At least **one UI button** to roll the dices (or enable the random results' generators scripts).
   - A **UI Credits** text object, which displays the Credits. Credits should contain all the 3D Objects', Music's and other components' sources used in your Project.
   - At least one **Camera** exists in the scene.
5. **Audio (5%)**: At least one **Audio source** playing either an ambient sound or a sound effect when something happens (ex. turn changes).

3.2 Extra Requirements(50%) + Bonus(10%)

You must complete the implementation of minimum requirements first, to start the implementation of the extra requirements.

1. **Graphics(15%)**: In addition to the minimum requirements, you should search third-party 3d models (found on the Internet for example from Sketch-fab or Unity's Asset Store) or create your own 3d models in separate 3d-modeling software (such as 3D Studio Max, Blender, etc) and import them in your project to build your scene. You must complete the implementation of minimum requirements first, to start the implementation of the extra requirements.

   You should create a scene with a board that consists of a path of nodes (at least twenty-five), plots (at least seven), hotels / buildings (at least four on each plot), and pass-through points (at least one gain money point and one buy entrance points), players (at least two) and two actual hexahedral dices (one for movement and one for build). Extensions of hotels are not required.

2. **Game logic and mechanics (25%)**: In addition to minimum requirements, your project must be able to provide the following functionality:

   Multiplayer Mode: The game is played in turns. Players move on a path that consists of four types of nodes **(Buy, Build, Free Build, Free Entrance [2.2])** and two types of pass-through points **(Gain Money and Buy Entrance [2.4)**. Players must be able to move on the path, buy plots, build buildings on them, place entrances on nodes, pay if they land at an opposing player's entrance, get a certain amount from the bank when passing the gain money point and buy entrances when passing the buy entrance point. In addition, you must create two actual 3d dice objects (one for **movement [2.1]** and one for **build [2.3]**).

   Note 1: Regarding the functionality of the **Build Node [2.2]**, your game logic does not need to include the player's ability to declare how many buildings the player wants to build, provided that the player has the ability to build only one building at a time.

   Note 2: Regarding the functionality of the **Buy Entrance Point [2.4]**, your game logic does not need to include the players' ability to declare how many entrances want to buy, as long as they can buy only

one entrance at a time.

Note 3: Your game logic does not need to include the players' ability to add an extension to his/her hotel.

Note 4: Your game logic does not need to include the players' ability to sell his/her hotels to another player (in case of player has not enough money to pay for some reason).

3. **UI & Cameras (10%)**: In addition to minimum requirements, your project must be able to provide the following UI functionality:
   - Seperate UI text objects presenting useful information about the players, such as **money**, **owned buildings** etc..
   - A **UI Credits** text object, that is enabled or disabled by a button, which displays the Credits. Credits should contain all the 3D Objects', Music's and other components' sources used in your Project.
   - One **Game Over UI** text object, that informs about the player who won.
   - At least two **Cameras** (which are changed by a button) in the scene.
4. **Bonus (10%):** Extended/Extra functionality and personal touch.

4 Basic Instructions

- Use your imagination and creativity to create a good-looking scene and scenario.
- You must complete the implementation of minimum requirements first, in order to start the implementation of the extra requirements.
- You can use Unity's **3D models** (cubes, spheres, ..) to create your scene by transforming them to give them the shape you wish.
- You can create your own 3d models in separate 3d-modeling software (such as 3D Studio Max, Blender, etc) and import them in your Unity project. However, if you don't have previous experience with 3D modelling, this option is not recommended.
- You can import third-party 3d models found on the Internet (Sketchfab) or Unity's Asset Store to help you build your scene. Remember: If you download and import third party 3D models you should include them in Credits.
- You can use the Built-in "Collaboration" tool in Unity to use for version control in your project (similar to Github).
- You must add appropriate **materials** to object surfaces to give them a realistic feel and look. Use Unity's Material Editor accordingly.
- It is highly recommended to use the Unity Documentation as it provides both explanations on all of Unity's tools as well as examples on how to use them. When you open the Documentation remember to select your Unity Version on the top left.

5 Physics

5.1 Collision between objects

Objects must have **Collider** components attached to them, otherwise they will pass through each other. It is important that the collider matches the 3D shape of an object as best as possible. For example, a box collider is ok for a box-shaped object, such as a wall or a smooth floor. More complicated 3D models require different types of colliders. For example, using a box or a sphere collider for a pin is wrong, as it will result in un-realistic behaviour.

Documentation - Collision between objects

5.2 Rigidbody

Rigidbodies enable your GameObjects to act under the control of physics. The **Rigidbody** can receive forces and torque to make your objects move in a realistic way. Any GameObject must contain a Rigidbody to be influenced by gravity, act under added forces via scripting, or interact with other objects through the NVIDIA PhysX physics engine.

A rigidBody is defined by many parameters, such as its mass, drag, angular drag and more.

Documentation - Rigidbody

5.3 Physic material

The Physic Material is used to adjust **friction** and bouncing effects of colliding objects.

To create a Physic Material select *Assets, Create , Physic Material* from the menu bar. Then drag the Physic Material from the Project View onto a Collider in the scene.

Documentation - Physic Material

5.4 Realistic physics

Game objects affected by physics (rigidbodies) have parameters, such as mass, drag and angular drag. Set their values accordingly by experimenting, so that they behave realistically.

6 Scripting: Game logic and functionality

Scripts control the entire functionality of your game, such as handling user input, triggering events upon specific actions, keeping score, moving objects and more.

Your scripts will have to:

- **Initialize** variables, load game objects and/or other scripts that need to be accessed.
- **Keep track of the game state** and act accordingly. Check for key presses, position changes, collisions and more...
- **Reset game objects' state** (such as position, rotation, gravity, ...) **when required**.
- Update the UI elements when necessary. For example, have a UI text that shows the current gold the player has. (Even better, also change the text color if the player has/ has not got enough gold to build another tower).

Important things to remember:

Code in Unity is written in C#. Every new script **derives** from the *MonoBehaviour* base class by default. A script deriving from *MonoBehaviour* enables the call of the Start(), Awake(), Update(), FixedUpdate(), and OnGUI() functions and has the following properties:

- A script will never be executed, unless it is attached to at least one game object in Unity. Furthermore, a single script can be attached to one or more game objects. In this case, **be careful**, as the same script will run as many times as the number of game objects it has been attached to.
- One GameObject can have multiple scripts attached, too. In this case, **be careful** of possible race conditions. (a race condition can occur

when, for example, 2 different scripts try to move the same object at the same time).

- Game objects (or other scripts) need to be initialised in the Start() function of the script before use, unless the script is attached to them, in which case they can be accessed directly.

The links contain useful tutorials on the communication between scripts and Game Objects in Unity:
Link$_1$ *Link*$_2$

7 UI , Audio and personal touch

7.1 User Interface (UI)

Unity's UI system provides a variety of tools to render elements, such as text or images, on the screen or at a specific position in the 3D space. The UI system can be used for many different purposes.

Documentation - User Interface

It is totally up to you to choose the design, position and functionality of the UI system - as long as it meets the minimum requirements.

7.2 Audio

You can use Unity's audio components to play sounds in a loop(ex. background music) or if a condition is true.

Documentation - Audio

7.3 Extended/Extra functionality and personal touch

This section is intended to give you the freedom to add your **personal style** to your game and add more advanced functionality.

Extra functionality may include:

- AI opponents.
- More than two players.
- More hotel-buildings.
- Adding particle systems for visual effects.
- Adding decorative objects and effects for a better looking scene.
- Animated objects.
- Different boards.
- Anything else you wish and it is not mentioned here.

8 Project Guidelines

- Your code must be organized in different scripts, depending on the functionality. **Each script must have a distinct role.**
- Use script names that make sense and attach them to related game objects. For example, create a "UIManager" script to handle the game UI, a "GameManager" script to hold variables and objects that keep track of the game's state.
- Your code must be written in C# programming language ONLY.
- Organize your game objects in a hierarchy that makes sense and is easy to understand. For example, if you have game objects such as "Cube(1)", "Cube(2)", "Cube(3)", ... , make them children of a single "CUBES" game object.
- Organize your asset files (textures, 3d models, materials, scripts, sounds, ...) in separate folders. Obviously, the name of the folder should be representative of the content.
- You may use the Internet (YouTube, blogs, ..) to your help for tutorials and code examples, but copy-pasting entire code or functionality is obvious and will NOT be accepted.

and remember while doing this project to **have fun !**