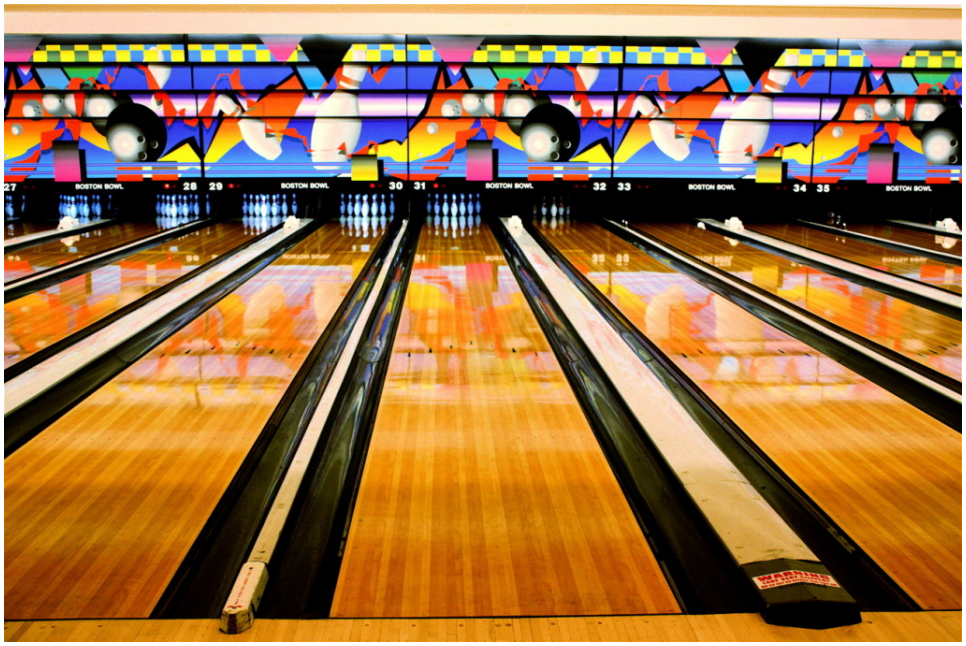
# 2nd Graphics Project: 3D Bowling Game

Unity3D

Panagiotis Drakopoulos - Scientific Responsible for Graphics Course



## 1 Introduction

The task is to create a realistic 3D Bowling game, using the **Unity3D** Game Engine, by applying the techniques demonstrated in class. A fully completed game (100% of marks) should consist of the following sections and components:

1. **Graphics**: bowling scene design, 3d objects, textures, materials, lights
2. **Physics**: colliders, physics materials, rigidbody

3. **Scripting**: game logic and mechanics, event handling, aiming and ball throw mechanism, camera angles

4. **Audio**: sound effects

5. **UI**: score display using Unity's UI components

The more sophisticated the implementation of each of the above sections, the more marks your project will gain.

The percentages below indicate an approximate division of marks for each section of the practical work. The division of marks should also guide the development effort:

- Game design (Graphics, UI, Audio) : **35%**
- Game logic and functionality (Physics, Scripting): **50%**
- Extended/Extra functionality and personal touch: **15%**

## 2 Minimum Requirements

1. **Graphics**: A bowling area with the appropriate 3d objects (floor, walls, balls, pins, ..) with realistic look and dimensions. All surfaces must have materials with at least one basic (albedo) texture attached to them, as shown in class.

2. **Physics**: Objects with colliders and appropriate components, so that they behave naturally and as realistic as possible. For example:

   - Some surfaces may require a physic material so that they have less friction.
   - The pins and the ball must have realistic mass and drag values.
   - The ball must be thrown by using Unity's *AddForce()* function only.

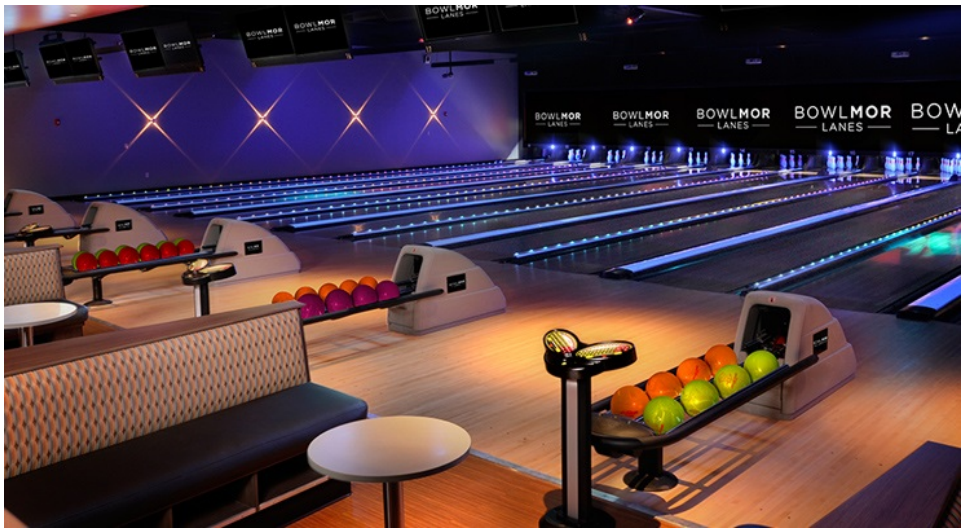
3. **Game logic and mechanics**: At the minimum, your finished project must be able to support a bowling game between 2 separate players-opponents.

   - Obviously, the game must follow the rules that apply on the real bowling game (they will be explained later). The game should start with the score as 0-0. As it progresses, each player will try to knock down as many pins as possible, to increase his score. When the game ends, it starts again and the score resets to 0-0.
   - You must implement some kind of aiming mechanism to throw the ball, using keyboard keys or mouse (or both). You can also use Unity's *"FPSController"* prefab, which has been demonstrated in class.

4. **UI**: At least two UI GameObjects, such as *Text*, in the scene to display the current game score and state, by using at least one *Canvas*. The *Canvas* could be in Screen Space and/or World Space mode. The UI components need to be handled by script, so that they display the current status or score of the game.

   **Note: All of the above have been demonstrated in class.**

# 3 Setting up the Scene



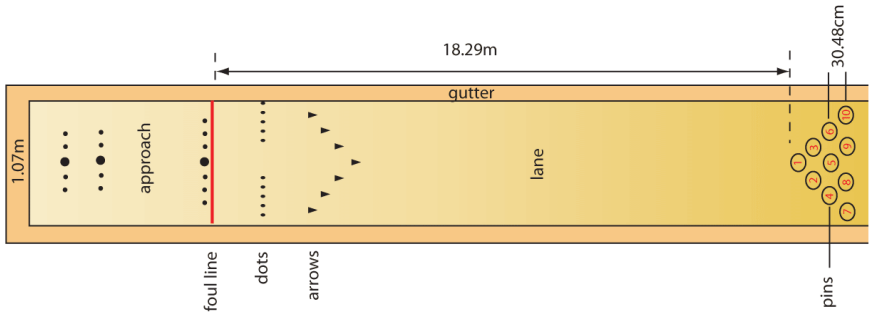

## 3.1 Basic Instructions

- Use your imagination and creativity to create a realistic and good-looking bowling scene. Take a look at real pictures for inspiration.
- You can use Unity's **3D models** (cubes, spheres, ..) to create your scene by transforming them to give them he shape you wish.
- You can create your own 3d models in separate 3d-modeling software (such as 3D Studio Max, Blender, Sketchup) and import them in your Unity project. However, this is NOT recommended if you don't have plenty of time and haven't used such software before.
- You can use third-party 3d models found on the Internet to help you build your scene (for example a chair, a table, ..).
- You may NOT use an entire pre-made bowling area.
- You can have as many bowling lanes as you want.
- Add appropriate **materials** to object surfaces to give them a realistic feel and look. For example, use Unity's Material Editor to make a surface look shiny/smooth or rough etc. `https://docs.unity3d.com/Manual/StandardShaderMaterialParameters.html`

- You can use extra **lights** to add style to your scene. Remember, you can customize the light color, position, intensity and more.

### 3.2 Dimensions

Either you make your own models or use pre-made imported ones, make sure their dimensions and scale are similar to the real ones. The image shows the dimensions of a single bowling alley for reference:




- A **pin** must be 15 inches (38 cm) tall and about 4.7 inches (12 cm) wide at its widest point.
- The maximum diameter of the **ball** is 8.595 inches (21.83 cm)and the circumference of the ball must not be more than 27 inches (0.69 m), and the ball cannot weigh more than 16 pounds (7.26 kg).

## 4 Physics

### 4.1 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.

`https://docs.unity3d.com/Manual/CollidersOverview.html`

### 4.2 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.

`https://docs.unity3d.com/Manual/class-Rigidbody.html`

### 4.3 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.

`https://docs.unity3d.com/Manual/class-PhysicMaterial.html`

### 4.4 Realistic physics

Game objects affected by physics (in this case: ball and pins) need to have realistic parameters, such as mass, drag and angular drag.

- The **ball** cannot weigh more than 7.26kg. Generally, the lightest ball available for use is 2.72 kg.
- The weight of a single **pin** must be at least 1.53kg and no more than 1.64kg.

As for the *drag* and the *angular drag*, set their values by testing and experimenting which ones work best. There is no specific correct value for these.

## 5 Scripting: Game logic and functionality

The scripting part involves the entire functionality of your game. That is from handling user input, triggering events upon specific actions and more.

Your scripts will have to:

- **Initialize** variables, load game objects and/or other scripts that need to be accessed.
- Apply force on the ball's **RigidBody**, according to the player's aim, by using the *RididBody.AddForce()* function, where RigidBody is the C# object that holds the *RigidBody* component of the ball.
- Constantly **keep track of the game state** and act accordingly. For example, if the ball has not yet been thrown, check if the user presses the buttons that throws the ball. If the ball has been thrown, pressing the same button should no longer apply any forces to the ball, (until the ball resets to the throwing position)
- **Reset game objects' state** (such as position, rotation, gravity, ...) **when required**. For example, it is a good idea to disable gravity on the ball, while on the aiming position, to prevent it from falling down. Additionally, the pins that where knocked down, need to be brought back to their original position for the next round.. and so on..
- Keep the **game score** and update the UI elements when necessary. For example, after each ball throw, you will have to find out how many pins where knocked down and update the player's score.
- Change the camera position/angle, depending on the game state. For example, when a player throws the ball, the camera follows it along the path, then it resets back to its original position.

**Important things to remember:**

- A script will never run, 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.
- Game objects can have multiple scripts attached to them, 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) that you want to access, need to be loaded in the Start() function of the script, before they can be used. However, Game Objects that are attached to the script, can be used directly and don't need to be loaded.

  The links contain useful tutorials on the communication between scripts and Game Objects in Unity:
  `https://unity3d.com/learn/tutorials/topics/scripting/how-communicate-between-scr`
  `https://unity3d.com/learn/tutorials/topics/scripting/getcomponent`

  **BOWLING GAME RULES**:
  `https://en.wikipedia.org/wiki/Ten-pin_bowling#Rules_of_play`

# 6 UI , Audio and personal touch

## 6.1 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.

In this project, you will have to utilize the UI system to display important game information, such as the current score and which player's turn it is to throw the ball. The most important UI components you will have to use is the **Canvas** and the **Text** components.

`https://unity3d.com/learn/tutorials/topics/user-interface-ui`

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.

When creating the UI, you will have to decide whether the Canvas will be in **World Space** or in **Screen Space** mode, or **both**.

If a Canvas is in World Space mode, the UI elements that it contains will be rendered at a specific location in the 3D scene, depending on the Canvas transform coordinates.

If a Canvas is in Screen Space mode, the UI elements will always be in front of the screen, no matter where the camera is or looks (like a HUD).

### 6.2 Audio

You can use Unity's audio components to play sounds in a loop(ex. background music) or after specific events (ex. ball collides with pins).

`https://docs.unity3d.com/Manual/AudioOverview.html`

### 6.3 Extended/Extra functionality and personal touch

This section is intended to give you the freedom to add your **personal style** to your game, extend or add extra functionality that is not listed in this document.

Extra functionality may include..:

- A.I. opponent.
- Menu
- A variety of sound effects
- Sophisticated aiming mechanism (for example giving the player the option to select the force of the ball throw..)
- Extra camera angles
- and more...

## 7 Project Guidelines

- Your code must be organized in different scripts, depending on the functionality. For example, use a script that handles the UI elemenets only (ex. menu, score display, ..), another script that handles the game logic (ex. throwing the ball), another script that handles different cameras positions and angles (if you include any) and so on.
- There is no specific/correct amount of scripts you should use, just make sure each script has a distinct role.
- 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 will NOT be accepted.

**Good Luck !**