German University in Cairo
Media Engineering and Technology
DMET 502 Computer Graphics, Winter 2025
Prof. Dr. Rimon Elias

Assignment 1
Deadline: Friday, 24th of October 2025 at 11:59 pm
Individual Assignment

Description & Game Play:

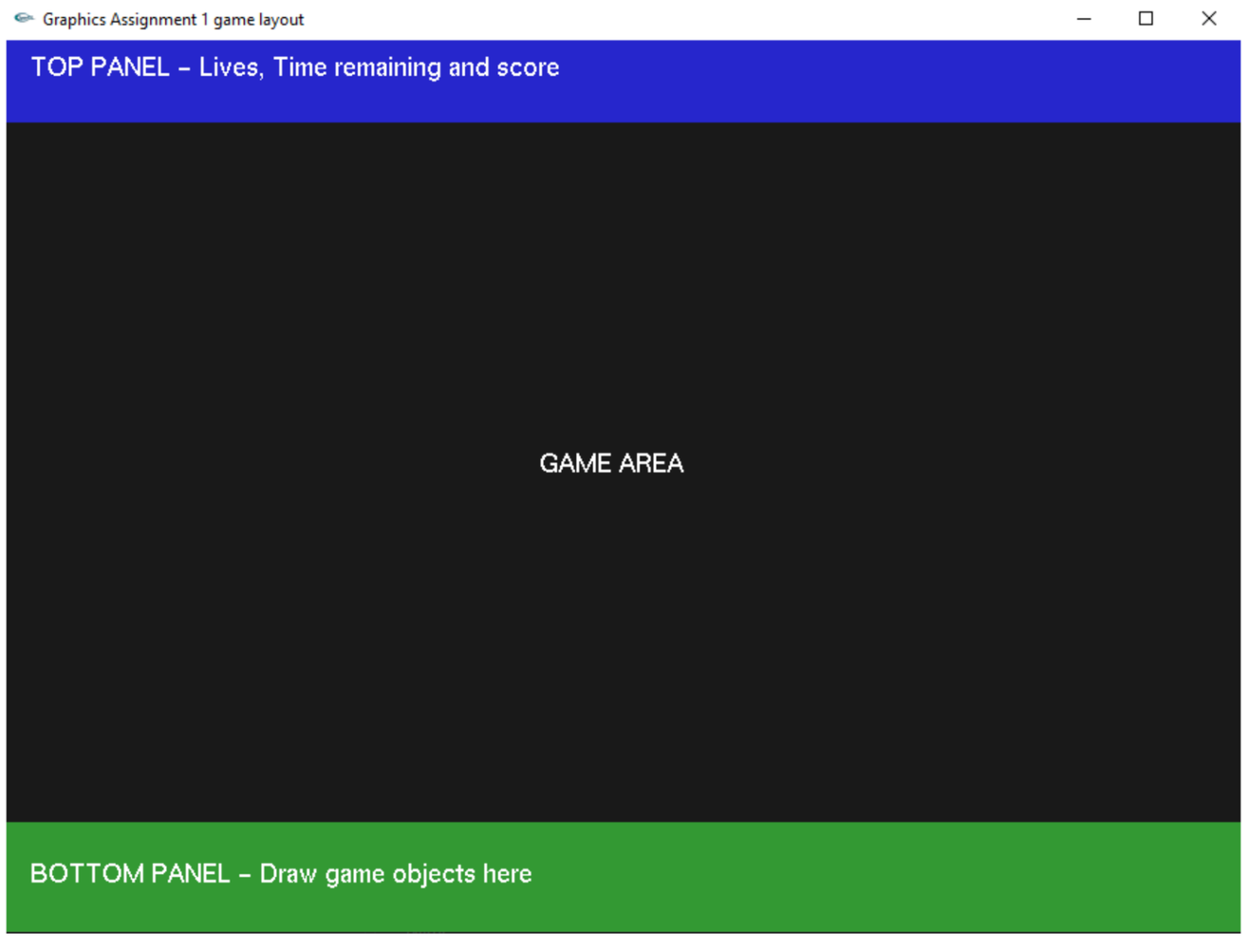
In this assignment, you are required to implement a 2D view game (top, front or side view). The main models in the game are the player, obstacles, collectibles, power-ups, and the game target. Collectibles increase the player's score, which initially starts at zero. The player has a total of five lives, which decrease when the player moves toward an obstacle. Power-ups are a special type of collectibles that have a greater positive impact on the player than normal collectibles.

The main objective of the game is for the player to avoid losing all five lives and achieve the highest possible score before collecting the game target within the allotted game time.

Choose a theme for your game for a better design and easier imagination. You should avoid having any religious, sexual, or political aspects in your project.

Environment:

Implement a 2D game using OpenGL and C++, where the game window has an upper and a lower panel. The upper panel has the health (displayed non-numerically) and time, and the lower panel has 3 shapes: obstacle, powerup, and collectibles. The game area is between both panels, where the player and the game target are within this area, as displayed in the figure below. panels, where the player and the game target are within this area, as displayed in the figure below.



German University in Cairo
Media Engineering and Technology
DMET 502 Computer Graphics, Winter 2025
Prof. Dr. Rimon Elias

Player:

The player and the game target are placed within the game area at opposite locations. For example, if the player initially starts just above the lower panel, the target will be positioned just below the upper panel. In another scenario, the player might start on the left side of the screen, while the target is located on the opposite side, such as the top-right area of the game area.

The player must collect as many collectibles as possible to increase their score, avoid colliding with obstacles, and reach the game target before time runs out. The player's movement is controlled using either the 'W', 'A', 'S', and 'D' keys or the arrow keys to move in all four directions.

Game Target:

The game target is defined in the position opposite to the player and animates horizontally or vertically depending on where it is initially placed. The animation is done in a Bezier curve continuous motion (Bezier code is attached with the assignment description).

A Bezier demo is available in the assignment .zip folder that shows how the target is moving during the game in a Bezier curve continuous motion.

When the player's position reaches the game target's position before the game time is up, the game ends and the player wins. If the time is up without reaching the target, the game ends and the player loses the game.

Obstacles, power-ups, and collectibles:

When the game runs, the scene is empty. The objects are placed inside the scene using the mouse clicks then the player can move within the scene collecting the collectibles and the powerups and avoiding the obstacles.

The user can use the left mouse button to place any of the shapes from the bottom panel into the game area (between the player and the game target). When the user left-clicks on an object in the lower panel, it activates the drawing mode for that object, allowing the user to add new instances of it within the game area. Once these objects are placed, their respective game logic becomes active.

For example, when the user selects the obstacle shape from the lower panel, the obstacle-drawing mode is activated. Each left-click within the game area will place a new

obstacle at the clicked location, which is then stored in the program so that its behavior applies (i.e., the player loses a life if they move into that location).

Similarly, when the user selects the collectibles shape from the lower panel, the collectibles-drawing mode is activated. Each left-click within the game area places a new collectible at the clicked location. When the player collides with a collectible, their score increases linearly, and the collectible disappears. For example, add 5 points for each collision with a collectible.

After all objects have been placed, pressing ‘R’ or ‘r’ on the keyboard starts the game and begins the countdown timer.

Power-ups:

Power-ups are special types of collectibles that can be added to the game area. Their functionality is up to your design choice, but each power-up’s effect should be temporary and end after a certain time. The game should include two different types of power-ups, each with distinct (different) functionality and appearance. Both types must also be available in the bottom panel for placement.

Modeling:

The shapes drawn onto the screen are described in this section. Note: primitive types include shapes found in Lab 1 pdf, such as point, line, line strip, etc. Some models would include different primitives, which means that you can not use the same primitive type twice.

- The **upper and lower game** panels should have **at least 1 primitive** each to draw the panels.
- The **player** is drawn with **at least 4 different primitives**. For example, a polygon is different from a triangle or a line. The **four different** primitives could be a circle, a line, a triangle and a point.
- The **obstacles** are drawn with at least 2 primitives.
- The **collectibles** are drawn with at least 3 primitives.
- The **two power-ups** are drawn **differently** with at least 2 primitives each.
- The two types of powerups, collectibles, and obstacles should appear **different** from each other.
- **Health** should be drawn with **at least 2 primitives** within the top panel in the game. For example, two circles inside one another, or two quads per health shape.

German University in Cairo
Media Engineering and Technology
DMET 502 Computer Graphics, Winter 2025
Prof. Dr. Rimon Elias

The requirements:

- **The environment:**
  - The top panel displays the player's health, score, and game time.
  - Health should be shown non-numerically (e.g., using icons or bars) and updated whenever a life is lost.
  - The bottom panel contains one object representing each game element — obstacle, collectibles, power-up type 1, and power-up type 2.
  - The game area, located between the two panels, holds the player and the game target at opposite positions.
  - Initially, the game area does not contain any other objects.
  - Game objects are placed by the user using left mouse clicks.
  - Pressing 'R' or 'r' starts the game, initiating the countdown timer.

- **Obstacles:**
  - **The mouse left-click** on the obstacle shape in the bottom panel activates the obstacle drawing/adding mode.
  - When the user left-clicks within the game area, a new obstacle is added at the clicked location (placed at the cursor location in the window).
  - Adding obstacles outside the game area is **not allowed**.
  - Placing obstacles on top of each other is also **not allowed**.

- **Collectibles:**
  - Left-clicking the **collectibles shape** in the bottom panel activates the **collectibles drawing/adding mode**.
    When the user left-clicks within the game area, a new **collectible** is placed at the clicked location.
    Adding collectibles **outside** the game area is **not allowed**.
  - Placing collectibles **on top of each other** is **not allowed**.

- **Power-ups:**
  - The game should include **two distinct power-up types**, each with a **unique functionality**.
  - When the player acquires (collides with) a power-up, it should **disappear** and its **effect** should **activate temporarily**.
    The effect of a power-up should last only for a few seconds, then **deactivates automatically**. For example, one powerup can speed up the player movement. The speed up lasts for a few seconds then the player returns to normal speed movement.
  - Power-ups with effects that do **not expire over time** (such as permanent health gain or increased game time) **should not be used**.

- **Player:**
  - The player's movement in **all four directions** is controlled by either the 'W', 'A', 'S', and 'D' keys or the arrow keys.
  - The player **cannot move** outside the game area boundaries (**or** outside of the window).
  - Whenever the player encounters an obstacle head-on, it should cost one of the five hearts and **the motion is blocked** while the obstacle **does not** disappear on collision.
  - When the player collides with a collectible, the collectible **disappears** and the score **increases**.
  - When the player collides with a power-up, the power-up **disappears** and its **temporary** effect is activated.
  - The player can miss collectibles or power-ups if no collision occurs (for example, if a power-up is above the player and the player doesn't move toward it).

- **Game End:**
  - **Game Win:** The player reaches the **game target** before **time** or **health** runs out.
    Game Loss: The player either **runs out of time** before reaching the target or **loses all lives**.
  - In both cases, the game area should transition to a **"Game Win"** or **"Game Lose"** screen, displaying the **final score** (displaying text function is included in bezier code attached with the assignment description).

- **Animations:**
  - **Player:**
    - The player moves in four directions and should rotate to face the direction of motion.
  - **Power-ups/collectibles:**
    - Power-ups/collectibles should be animated in position (any sort of animation, rotation in place, scale up and down, or translating up and down or left and right). Animation is a continuous animation (does not depend on a key press or mouse click).
    - Power-ups/collectibles should have **different** types of animation. For example, if the collectibles animate using translation, the power-ups should not animate using translation.
    - Both powerups should animate and their animations **can be the same**.

- **Background:**
  - There should be some type of animation in the background other than the health, score, and time updating throughout the game (moving background). Animated background is implemented by having continuous movement of an object by applying continuous translation, rotation or scaling up and down.

- **Game Target:**
  - The game target should change its position within a range in either the x-direction or the y-direction in a bezier motion with time.

- **Bonus:**
  - **Sound:**
    - Background music that starts playing when the game starts,
    - A sound effect plays when collecting objects from the scene.
    - A **different** sound effect plays when colliding with obstacles in the scene.
    - A **different** sound effect upon game win.
    - A **different** sound effect upon game loss.
  - **Textures:**
    - Apply textures to everything in the scene except for very small objects, where texture images will be invisible.
  - **Note:**
    - The bonus is acquired by **EITHER** sound (all points) **OR** texture.

Submission:

- The assignment should be implemented in **OpenGL**
- This is an **INDIVIDUAL** assignment.
- This assignment is worth **7.5%**
- **Deadline** for the assignment: **Friday, 24th of October 2025, at 11:59 pm**.
- Cases that will lead to a **ZERO**, includes:
  - Copying the code from the **Internet** or **GitHub repo** or a **colleague**.
  - You are expected to fully understand **any** code generated by **ChatGPT**, as it is part of your assessment.
  - Files with **compilation errors** will not be graded. It is zero.
  - During the evaluation, you will answer questions based on your implementation. **Not answering questions, will result in a deduction and can lead to zero.**
- Submission guidelines:
  - You should name your **.cpp file** in the following format PXX-58-XXXX,
  - **.cpp file only to be submitted, no assets or other files.** However, if your code is divided into multiple **.cpp files and .h files, submit them in one archive.**
  - **Do not** submit the **.sln** of your assignment, to avoid getting a **zero**.
  - Submission form: posted on CMS.