Exercise 5: Freestyle

IMPORTANT

- To aide in peer reviewing, please be sure to implement your game in our standard version of Unity: 2020.3.18f1 LTS.
- Be sure not to more than 10Mb worth of audio files

Scope and intent

For this assignment, you will make a simple arcade-style game of your own choice. This isn't an assignment about game design, or about any new programming concepts. It's about making a game starting from a blank slate. So the two main ways to fail at it are:

- Wait until the last minute to start and then run out of time
- Over-scope: plan to do too much and then end up with lots of partially implemented features and no completely implemented features

So start early and be realistic. Think 1970s and 1980s games, not *Cuphead*. And try to get to a minimum playable game as quickly as possible. You can always add features later.

Acceptable genres

You can make any single-player, 2D game that satisfies the criteria below, so long as it isn't a reimplementation of one of the games from the previous assignments. Examples would be:

- A **platformer** (running and jumping).
  If you do this, use the real physics engine rather than faking it like we did in the platformer assignment. You would also need to have other objects moving around in the world besides the player.
- Collision-based physics games: **pool**, **pinball**
- A **fixed shooter**¸ *a la* Space Invaders or Galaxian
- One of the simpler **classic games** such as, Breakout, Missile Command, or Frogger

Unacceptable genres

Don't do Asteroids or Angry Birds (at least not something that's exactly like the Angry Birds we already did). Don't do Pong or a visual novel, since they don't satisfy the requirements.

If you can't decide what to do

If you can't decide what to do, I'd do one of these:

- A **fixed shooter**
  You're on one end of the screen and can move one-dimensionally and shoot. The enemies start at the other end and move toward you. If they get past you, something bad happens (like losing).

- Some variant of **pool**
  Player moves around and kicks balls through whatever control scheme you choose. Those balls hit other balls and that somehow leads to scoring.

Requirements

You can make any kind of game you like provided that (this is roughly what the grading rubric will be):

- Objects are on the screen
  - The player must have some on-screen avatar that moves (their ship, their tank, a box, a little picture of a person)
  - There must be other on-screen objects (plural) too
    - At least one of those other objects must move around
  - There must be collision detection between the objects; they can't pass through one another
- Controls
  - The player must be able to move around in the game world using some control scheme
    - This can be keyboard control, mouse control, or controller control. Your choice.
  - The player must be able to take some kind of action beside moving (more than one action is fine)
- Scoring
  - The player must be able to score in some way
  - At least one action they can perform must somehow allow them to score
    - It's okay if there are other ways of scoring too
    - And it's okay if there are things that can reduce their score
  - Their score must be displayed on the screen
- Sound (see appendix)
  - Changes to the score must be accompanied by sounds
  - It's fine to include sounds in response to other events too
- Termination; either:
  - A timed game: the game runs for a specified amount of time and the player tries to accumulate the highest score they can, or:
  - A game with win/lose conditions:
    - There is a win condition
    - There is a lose condition
    - The game ends when a player wins or loses.
    - The player must be able to tell when the game ends and whether they won or lost
- Playability
  - The game's instructions must be sufficient to allow the reviewers to play it and understand if it's working
  - The player shouldn't be able to get into "stuck states" such as flying off screen and not being able to figure out how to get back

You must include instructions for your game. You are responsible for insuring that the reviewers understand the intended behavior of your game well enough that (a) they can play it and (b) they can determine whether you've successfully implemented the requirements listed above.

Writing the game

There's no starter code for this. Use what you've learned to make a basic game. If you need assets like sprites and sound files, google "free game sprites" or "free game sounds." There's a lot that's available. Or if you don't want to use sprites, you can use the polygon renderer from Exercise 1.

Academic dishonesty and intellectual property rights

You may use any sprites and sounds you like for this assignment provided that:

- You can legally use and distribute them (don't pirate them)
- You include file named CREDITS.txt that documents who their original authors were, or failing that, what web site you got them from.

There are a few things you're allowed to recycle from previous assignments, if you like:

- Any sprite or sound assets
- The ToonExplosion animation used in the angry birds game clone
- The `Polygon.cs` file from Exercise 1, which draws polygons on the screen. Using this means you don't have to find sprites, but means everything on screen needs to be brightly colored polygons
- The score display code from any of the previous assignments
- The menu code from any of the previous assignments (although menus aren't required for the assignment, so I wouldn't recommend trying to add them.

Apart from the above, and the Unity engine itself, you **may not use any other code unless it is written by you** personally, without written permission from Ian in advance. Not starter kits, no code or DLLs from the Asset store, and nothing from github, etc. You are free, however, to consult on-line sources for discussions of general implementation techniques and to copy those techniques.

What to turn in

Upload a ZIP file to canvas containing:

- An instruction file called INSTRUCTIONS. If it's in some format other than Word or TXT, include a PDF version of it. The file should make very clear
    - What all the objects on screen are
    - What their behavior is supposed to be
    - What the player's controls are
    - How the player scores
    - How the game ends, including win/lose conditions, if appropriate
- Your CREDITS.txt file, if you used assets you obtained on-line
- Your Assets, Packages (if any), UserSettings, and ProjectSettings directories

Be sure your ZIP file is under 20MB.

Appendix: How to add sound to a game object

You can add sound to a game object using an AudioSource component and one or more AudioClip assets.

- Add an AudioSource component to the game object
- You will need a script (i.e. component you wrote) to initiate sound playback whenever it is appropriate. So if you do not already have an appropriate script attached to the game object, add one now.
- You will need one or more sound file(s) for the sound(s) you want to play. Drag it/them into the Assets folder. Unity will treat them as AudioClip assets.
- The script that is to play the sound must have access to the AudioClip (the sound file). So add public fields to its class of type AudioClip. These will appear in the Unity editor.
- In the unity editor, drag the appropriate sound file(s) into the AudioClip field(s) of your script.
- In the code for the script, add calls to the PlayOneShot() method of the AudioSource in the appropriate event handlers. For example, if you want to play a sound when two objects collide, you would add a call to PlayOneShot in the OnCollisionEnter2D method. Remember this is a method of the AudioSource component in your game object, so you may want to use a Start method to store the AudioSource in a field so you don't have to keep calling GetComponent all the time.