Due October 13th at 11:59pm

Requirements:

- Create a 3D game named *'Kitchen Shooter'*. The game should meet the following requirements:
  - First, watch the 2 demo videos posted on Canvas. Basically, your game must look and work like the one in the videos.
  - Import asset files posted on Canvas (`kitchen_shooter.zip`) into your game.
  - Use 16:9 Aspect Ratio on Game View.
  - First put up the Kitchen image on the Background Wall.
  - Player (Cannon) moves horizontally and vertically via user's Left/Right/Up/Down key press. The Cannon is always centered on the screen, which means Camera must follow Cannon's movement.
  - Camera's Viewing area (and thus Cannon) should never go out of bounds of the Kitchen background image, that is, its movement must be bounded on all 4 sides (see video). So user never sees beyond the edges of the Background Wall.
  - Generate 10+ teapots by importing `teapot.fbx` file. Make sure they are all rendered with metal-like shiny material (see video).
  - Each teapot must be slowly rotating about y-axis all the time. Make sure their starting rotation angles are all different (so their rotations are not synchronized).
  - When user presses Space key, Cannon fires a Ball, whose surface also has shiny (but darker) material.
  - If a Ball hits a Teapot, the hit Teapot must fly out of its position, then possibly hit the Background wall, then fall down (by gravity) and disappear. A hit Teapot must stop rotating. *Important Tip: Do not use Mesh Collider for Teapot, as Mesh Collider works for static objects only.*
  - When a hit Teapot hits another Teapot, the latter Teapot must also fall down and disappear. Thus, you can possibly kill multiple Teapots in one shot.
  - If a Ball misses all Teapots, it must then hit the Background Wall and then fall down and disappear. Make sure the fired Ball is destroyed once it is out of screen (to avoid clogging up the memory).
  - Display Timer (UI Text) that counts down from 30 seconds (must be updated each second). Use Font *BradBunR* included in the asset package.
  - The object of the game is to kill all Teapots within 30 seconds.
  - If time runs out before hitting all Teapots, display *Game Over* and stop the clock.

  - If User successfully hits all Teapots within 30 seconds, display *You Win* and stop the clock.
  - You may adjust the number of Teapots as you see fit.
  - Make sure to add *Restart* button to let user restart once the game is over. (*Missing restart button will lead to point deduction*)

- Add one more twist of your own to the game, and be creative.
  - For example, you may introduce a special evil teapot which, if hit, adds 5 more teapots to the scene. But do not use this twist, use your own. :-)

- Finally, build your game as follows:
  - First, create *Build* folder under your Unity project directory.
  - Select *Windows* as your target platform (even if you're using Mac).
  - Click *Build* and select Build folder you just created.
  - The build file (.exe) must be created under Build folder. (*Missing build file will lead to point deduction*)

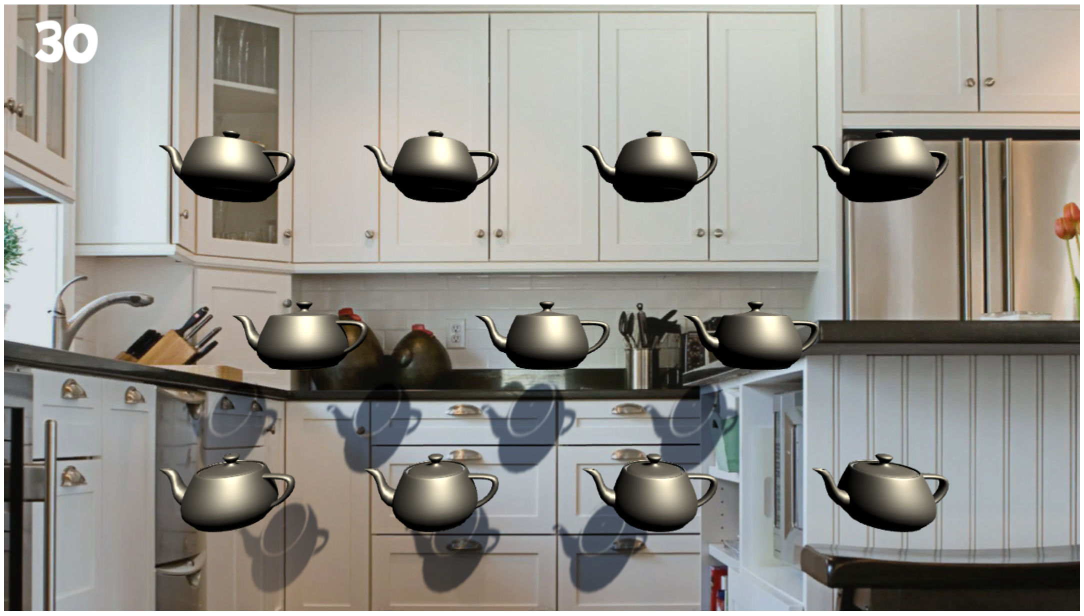

Figure 1: Kitchen Shooter

What to submit:

- Your entire Unity Project directory (containing all assets and build files) must be submitted.

How to submit:

- Use Canvas Assignment Submission menu to submit the assignment electronically at Canvas.
- Make sure to zip your entire Unity Project directory into `hw3.zip`, then submit your `hw3.zip` as a single file.
- When submitting, add comment on Canvas for grader describing what special twist you added to the gameplay. (*Missing description will lead to point deduction*)

Policy

- At the top of each Script file (.cs), provide comments specifying the author, date, and a brief description of the file.
- Each Script file (.cs) must contain enough comments here and there to make it easy to follow your code. Insufficient comments could lead to point deduction.
- Incomplete project will get almost no credit (e.g., game does not run due to script errors or game terminates prematurely due to run-time errors).
- *Thou shall not covet thy neighbor's code*. If identical (or nearly identical) submissions are found among students, every student involved will get automatic zero for the assignment. The same goes for copying existing code from online source.
- If a student makes multiple submissions, only the last submission will be considered valid.