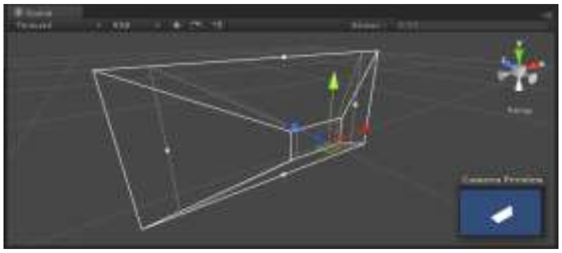
# CORNELL NOTES – COMPUTER PROGRAMMING & GAME DESIGN I

| | | |
|---|---|---|
| | *Topic/Objective:*<br>***Level 4:***<br>***Lights, Camera, Actions!*** | *Name:*<br>*Class/Period:*<br>*Date:* |

*Level Objective:*
***To add items to your video game that help make it more visual appealing and easier to play with including cameras, lighting, and character controllers.***

| *Questions:* | *Notes/Answers/Definitions/Examples/Sentences:* |
|---|---|

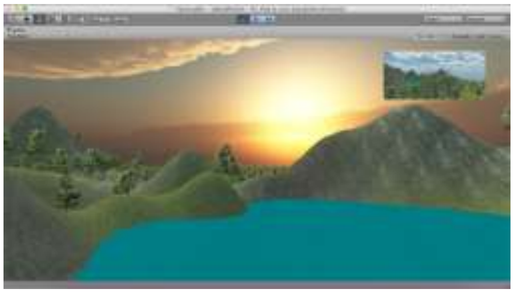
**Working with Cameras**



- The camera is the player's **view into the world**
- Creating a new camera (**GameObject > Camera**)
- The white outline represents the view that the camera can see (called the **view frustrum**; see image above)
- Cameras (can move and rotate them with their respectively tools)
- You can have **multiple** cameras (Ex. **for multiple characters in game**)
- Depth – **Specifies the priority for multiple cameras**
- A scene can only have **one single audio listener**
  - When adding multiple cameras, **disable** the audio listener

**Advanced Camera Stuff**

- Adding cameras to a 3D Object (i.e for having a multi-player game)
  - **Inspector Window > Add Component > Rendering > Camera**
- What if we have multiple cameras in our scene (which one will display when we play it?)? → The Camera with the greatest **Depth** in the **Inspector Window**
- Split Cameras
  - Change Depth to **-0.1** for each camera
  - Viewport Rect – **Changes the size & position of camera when played**
    - Keeping width/length at 1 = **camera takes up entire screen**
    - Keeping Y at 1 = **camera will show at top part of screen**
- Picture in Picture (PIP) Effect
  - **Camera takes up whole screen with another mini camera showing (i.e. camera looking behind with your car)**
  - Camera #1
    - **X=0, Y=0, W=1, Y=1**
  - Camera #2

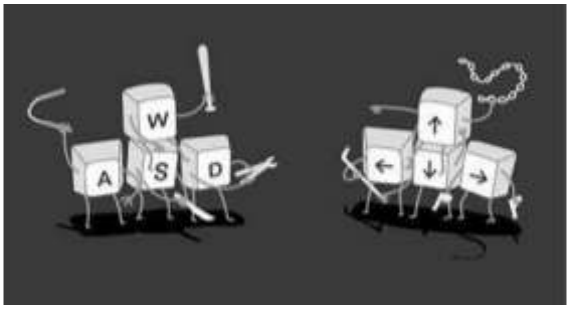


Would vary (Example: **X=.75, Y=.75, W=.20, Y=.20**)

# CORNELL NOTES – COMPUTER PROGRAMMING & GAME DESIGN I

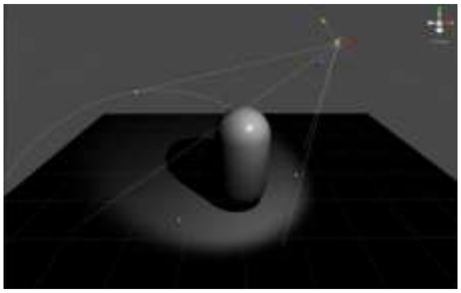
| *Questions:* | *Notes/Answers/Definitions/Examples/Sentences:* |
|---|---|

## Character Controllers (Action!)



- After adding your FPS Controller, make sure to **deactivate any cameras** that are currently running (**uncheck in the Inspector Window**)
- Unity will now automatically default to the **First-Person View**
- Download the Characters Folder in order to get Unity's FPS Prefab (included with the Standard Assets Package)
- **Drag your FPS Prefab into your scene** to where you want your character to start appearing at.
- We can now click **play** and view in **Game View**.
- By default, here are your controllers:
  - To look around → **use mouse**
  - To move up/down/left/right → **WASD keys**
  - To begin running → **SHIFT**
  - To jump → **Space Bar**
- After adding your FPS Controller, it's a good idea to walk/run through various portions of your **terrain to see if there are issues (ex. too steep).**
- Issues you might run across after playing your game:
  - FPS falls through terrain→ **position FPS above terrain (Move Tool)**
  - FPS bounces onto terrain → **FPS too high; position FPS on ground**

## Working with Lighting



- Lights go a long way in determining how a scene is **perceived**
- All new scenes have a **Directional Light** by default
- You can move and rotate them (with their respective Transform Tools)
- To add multiple lights, choose **GameObject > Lights > *You choose***
- Baking vs. Real Time
  - Real Time – **game engine calculates the light and shadow for every single frame (looks better but can affect performance)**
  - Baking – **precalculates light and shadow for textures & objects during creation (really nothing more than Lightmap textures; can make games run more quickly)**
- Light Styles We Can Use
  - Directional Light - **Points in specific directions (ex. sunshine)**
  - Point Light – **Light emits in all directions (ex. weapon glows)**
  - Spot Light – **Projects cone of light forward (ex. Street Lights)**
  - Area Light – **Rectangle where lights go in all directions (ex. Ceiling Light**