**Ambient Lights** : An ever present light glow to minimally allow object to be seen without shadows being cast – Imagine a room with perfect lighting where you don’t know where the light is coming from but you can still see everything – Mostly just shows basic background color and foreground color

**Environment Light** • Comes from a skybox, simulating the sun or moon in the skybox • Changed through Window->Rendering->Lighting Settings->Environment Lighting.

**Directional Light** • Similar to ambient light but there can be many of them and it casts shadows • A light with only a direction, no position – Light source is infinitely far away – Light rays travel in straight lines • Good for simulating sun or just generally brightening a scene from one side

**Point Light** • Light emits from a single point • Emits in a sphere in all directions around that point. • Light gets dimmer the further it is from the source • Good for simulating unconstrained light sources, like an explosion

**Spot Light** • Light emits from a single point • But comes out in a cone away from the light source • Light gets dimmer and disperses the further it is from the source • Good for constrained lights such as ceiling lights and lamps

**Emissive** • Light is emitted from the color properties of an objects material (see later slides) • Good for glowing object (e.g. neon signs)

**Material** • Meshes only specify the surface **shape** of an object • Materials specify the **detail** on the surface of the object – Colour – Texture (e.g. patterns on the surface) – How light bounces off the objects • Sperate assets in the Project Window – Attached to Renderer components on game objects in the scene • Shader – code that dictates how a material is applied to a mesh – And additional effects once it has been applied

**Normal Maps** – specifies which direction each (u, v) point is facing to determine how light will bounce off the 2D image – “Bump mapping” – Normal maps allow for bumps and grooves to be added and react naturally to lighting – Made more intense and realistic through Height Maps

**Week 10**

Canvas

•**Everything in a UI is placed under a GameObjectwith a Canvas component**

–I.e. all UI elements are a child of a Canvas

–Will be auto created if any other UI element is created and one doesn’t exist.

•**One scene can have multiple canvases**

–Best to minimize quantity where possible

•Creating a canvas will also create an EventSystem.

–Handles UI Input

–Mostly can be ignored for basic UI’s

•The canvas covers the **entire screen**.

–I.e. has the same dimensions as the screen it is viewed on.

•Has no depth

–Everything is flat (no perspective)

–Z coordinate is meaningless

–Rotation around y axis is meaningless

•The canvas covers **one camera.**

–Has same dimensions as camera view.

•If the camera is in Perspective mode

–Then the canvas will have perspective.

•Otherwise quite similar to Screen space -Overlay

•Occupies a specific spot in **world space**.

–Just like every other game object in the scene

•Allows for embedding text and icons in the game environment.

•Warning: Because it is not an overlay, the canvas won’t be aware of different screen dimensions.

–Can lead to text that is either too small or that is blurry

–Not recommended to use as replacement to Screen Space -Overlay

Element Ordering

Canvas:

•Sort Order –Order of rendering for all Canvas gameobjects •Lower values **rendered first** (i.e. rendered behind)

•Within a Canvas: Objects higher in the Hierarchy Window are **rendered first** (i.e. behind lower objects) •Parent objects are **rendered first** (i.e. behind child objects)

**Interaction ordering**

If another UI gameobjectis lower in the hierarchy, it will receive the click and block the button.

•EXCEPTION –Child images/text will not stop a parent from being clicked, even though they appear in front of the parent. –If both parent and children have interaction (e.g. both are Buttons), the child lowest in the hierarchy will always block the others

**RectTransform**

UI gameobjectsdo not have a Transform component.

–They have a **RectTransform** component (which is a sub class of Transform)

**Anchors and Pivots**

•Determines relative positioning

•**Anchor**–what part of the screen / parent object to attach to.

–If that part of the screen / parent moves, the object will move

•**Pivot**–Where the “centre” of the UI element is

–PosX, Y, Z, Width, Height, Rotation values all start at the pivot.

**Stretching**

•Splits the pivot into 2 or 4 and assigns it to the anchors

–No more PosX, Y, Z -> Now [distance from] Left, Right, Top, Down

•If any anchor moves, the RectTransformwill **move and scale**

**Screen Resolutions**

•Designing a UI for one screen doesn’t mean it will look good on another screen.

•Wide variety of Aspect Ratios

–Aspect Ratio = Width / Height = base width : base height

•Monitors

–1680x1050 = 16:10, 1920x1200 = 16:9, 1920x1080 = 4:3, 2160x1440 = 3.2, 3840x2160 = 12:5

•Phones (in landscape mode)

–iPads = 4:3, iPhone 4 = 3:2, Android tablets = 16:10, Galaxy Tab 7 = 17:10, iPhone 5/6 and Galaxy 6/7 = 16:9… and smaller brands with a lot more.