VR best practices and UI design

Jacopo Essenziale
Manuel Pezzera
Renato Mainetti

jacopo.essenziale@unimi.it
manuel.pezzera@unimi.it
renato.mainetti@unimi.it

Lab 11

Designing VR applications

ONE MAIN GOAL

MAXIMIZE USER IMMERSION

What can go wrong?

- Sensory conflict
  - Unnatural stimuli in the virtual world may lead to bad VR experience and sometimes to VR sickness
- VR Sickness
  - Some people are more sensible to sensory conflicts and may feel sick while using your VR applications
- User Safety
  - It is important to be concerned about user safety when designing VR application as while operating in the virtual world they are more vulnerable in the real world.

Sensory conflict

- Happens when perception of self motion is based on incongruent inputs from visual, vestibular and non-vestibular systems.

VR Sickness

- Sensory conflict appear to be one of the main cause of VR sickness.
- We should be aware of this problem when integrating locomotion inside our VR application

GOOD NEWS

- The more time users spend using VR, the less likely they are to experience discomfort, this is caused by our brain learning to reinterpret visual anomalies

How to minimize sickness issues

- **Always respond to user inputs**, even on menus, while game is in pause etc.
- **Do not instigate movements without user inputs** (e.g. rotate camera while user is not moving the head)
- Movement in Virtual World should be **consistent with head** and body movements.
- Think very well on how to implement locomotion in your application, the safest (even if harder) option could be to **avoid locomotion by design**

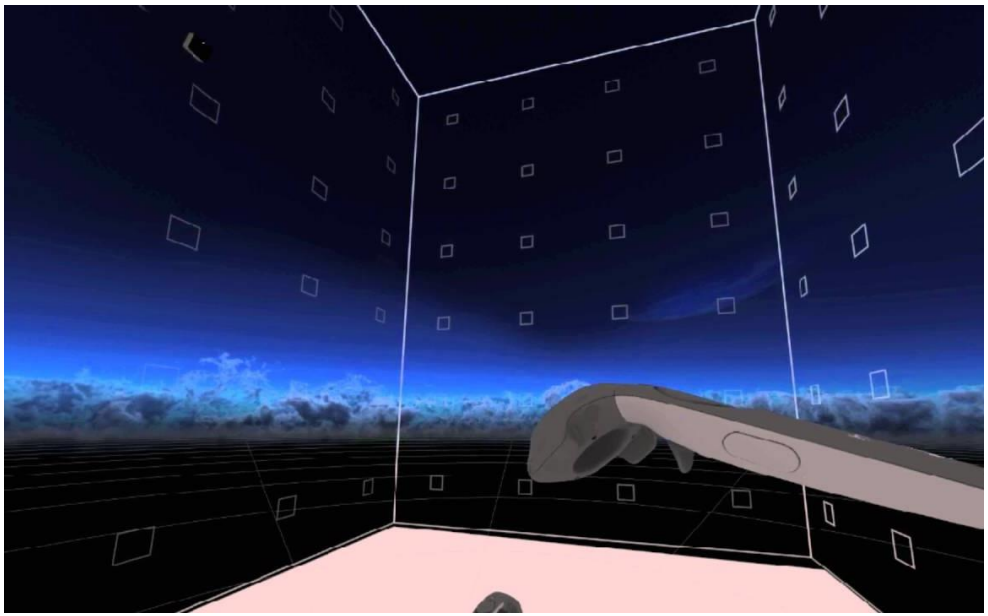
Mind users' safety

https://www.youtube.com/watch?v=0KcllPEe8y8

Mind users safety

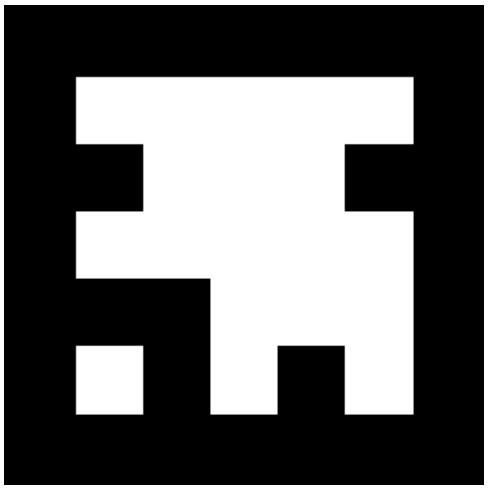
- Allowing users to freely move in the real world while blinded by a VR headset, tends to be a bad idea in general.
- However, finding a good solution to this open problem, would greatly improve VR experience.

HTC Vive solutions to this problem is to place some gizmos in the VR world that remind the users where are the boundaries of the safe movement area in the real world.

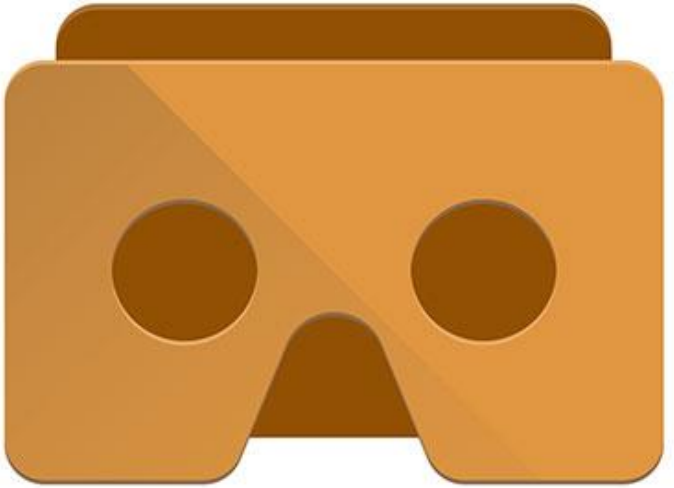


Project idea?

Using AR markers and libraries to locate the user in real world, could allow us to dynamically build a virtual world in which the user could freely move around.



+



Locomotion in VR

- Acceleration vs. Speed
  - While **acceleration** seems to be the **primary cause of discomfort** in VR experience, it appear there is **no straightforward relationship between speed and discomfort**.
- General guidelines:
  - Allowing user to set the pace of movement may help in control discomfort
  - Forward movement is more natural than backward or side movement
  - Open spaces with respect to enclosed spaces are known to be more comfortable to users.

Locomotion Techniques

- **No locomotion by design.** To completely avoid locomotion discomfort, is to design VR application that do not require locomotion at all. (When possible)
- **Teleportation.** Allow users to move around the scene without having to deal with accelerations.
- **Artificial movement.** Moving on a railroad or on a predefined path at a fixed speed.
- **Fade to black.** Similar to teleportation, fade to black before switching location.
- **HMD acceleration.** Use HMD accelerometers and gyroscopes to allow users to control movement direction. (e.g. airplane movement)

Interaction techniques (1)

- Gaze based interaction. (Staring at objects causes something to happen in virtual world)
  - Requires a UI element to show what exactly we are pointing to. (e.g. little dot in the middle of the screen)
- Dedicated controllers (e.g. Gear VR, Oculus or HTC Vive controllers)… usually 3 or 6 DOF
  - Good practice is to render the controller in the virtual world, with a ray coming out from it showing where the user is pointing at.
  - Changes in ray or target colors are encouraged to provide feedback on the presence of an interactable object

Interaction techniques (2)

- Hands (Leap motion controller):
  - **Rendering hands** in the scene may **greatly enhance VR experience**, however **render a whole avatar only** if you are sure that it will be **perfectly aligned** with the user real position.
  - Mind Hardware Limitation, leap motion work best when hands are right in front of the HMD, so design your applications to encourage user to interact with the world in this way. Also consider the Leap field of view when designing your app (Do not require users to look at one side and interact with an object placed far away from where they are looking)
- Other way? Custom Hardware? It's a good project idea!

Catching users attention

- This is particularly important in 360 videos, but meaningful also for real time applications. The main action in a video may happen in a place where the user is not looking at…

Catching users attention

- An example of use of good design practices can be observed here…

https://www.youtube.com/watch?v=BEePFpC9qG8

Catching users attention

How To?

- Lighting effects -> focus lights on the place you want the user to look at.
- Visual effects -> also particles, in-world UI elements (such as arrows), etc. may be used to focus on a particular element of the scene.
- Spatial Audio, information on where something interesting is happening in the scene may be provided by audio effects, spatialization will help the user find the source of the audio easily!

Optimization

- Frame rate is an essential part of ensuring users have a great and nausea-free VR experience, optimization is a critical part.
- Unlike some other platforms, it’s best to optimize early and often with VR.
- Testing regularly on the target devices is also essential.

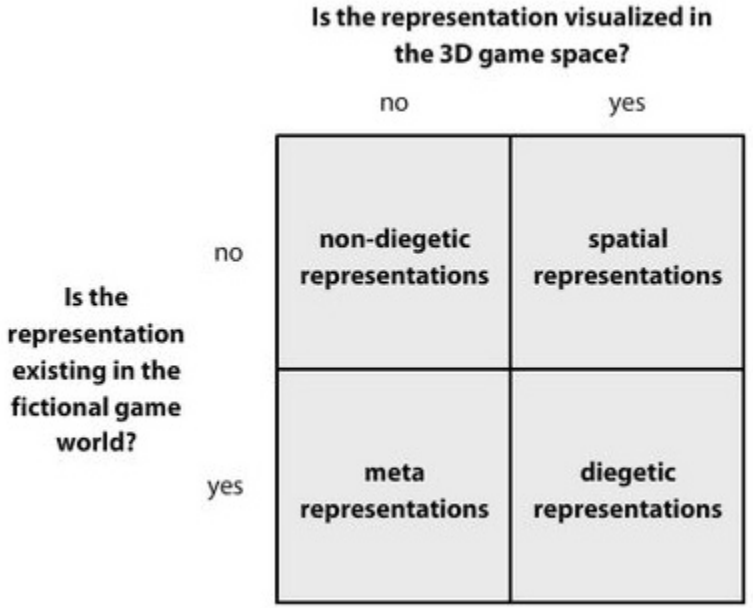
UIs

Diegetic UI: Interface that is included in the game world

Non-diegetic UI: Interface that is rendered outside the game world, only visible and audible to the players in the real world

Spatial UI: UI elements presented in the game's 3D space with or without being an entity of the actual game world

Meta UI: Representations can exist in the game world, but aren't necessarily visualized spatially for the player



UIs

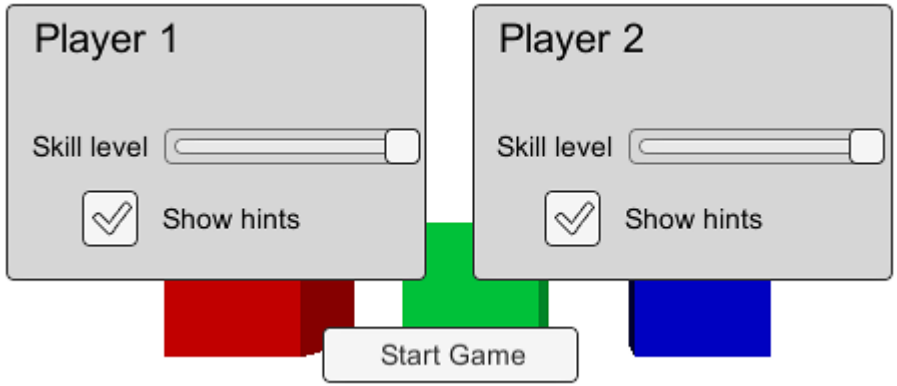

120
O
N
150
Obiettivi
Distruggi 30 nemici
Suggerimenti
Premi L durante il countdown se vuoi far la prossima orda in anticipo
Inventario
Infiniti
100%
Infiniti
100%

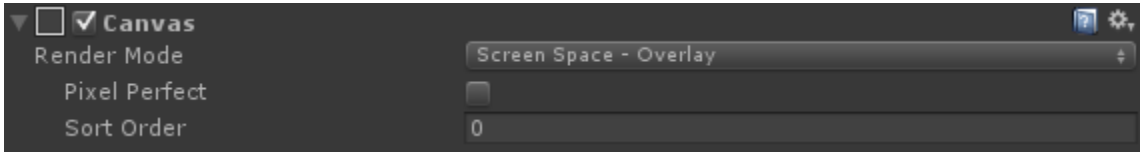
UIs in traditional games

- Called Non-Diegetic UI
- Used to show health, score…
- It does not exist in the world

UIs in traditional games - Unity

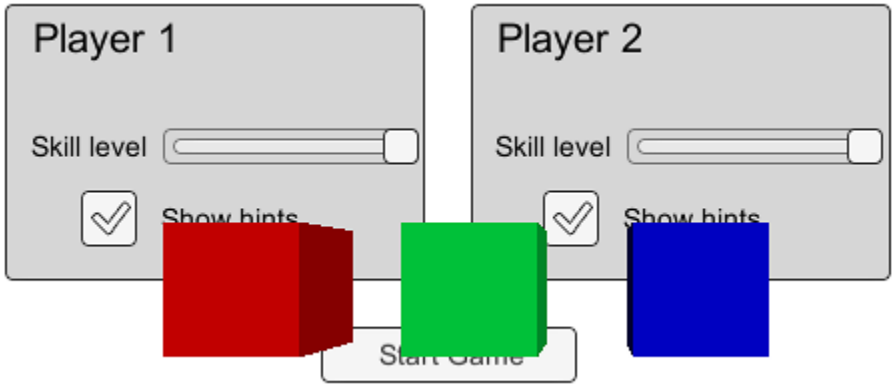
- Screen Space – Overlay
  - Canvas is scaled to fit the screen
  - UI drawn over any other graphics

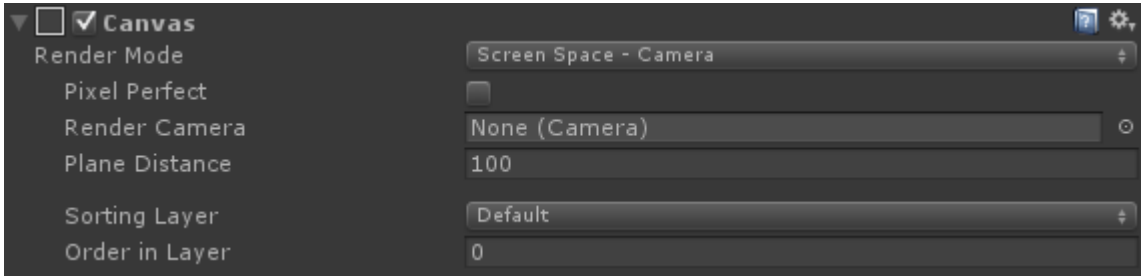




UIs in traditional games - Unity

- Screen Space – Camera Render Mode
  - Canvas is rendered as if it were drawn on a plane object some distance in front of a given camera
  - Objects that are closer than the plane will be rendered in front of the UI





UIs in VR

- Traditional UI does now work in VR
  - Our eyes are unable to focus on something so close
  - Screen Space-Overlay is not supported in Unity VR

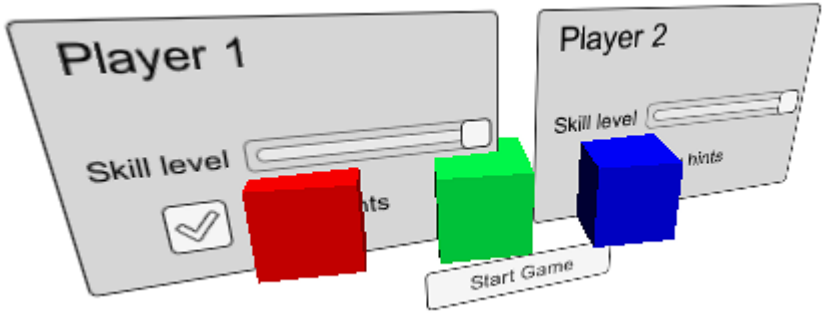
- Solution?
  - Diegetic UI or Spatial UI!

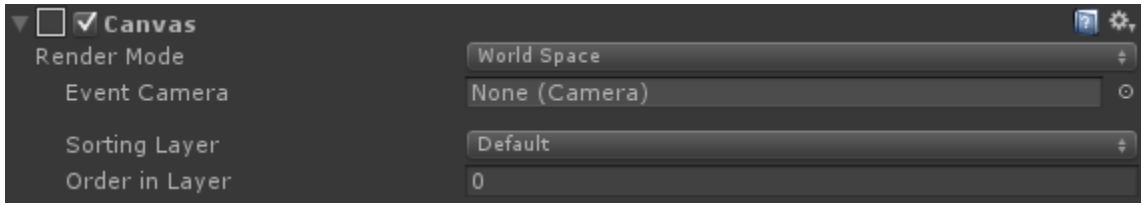
Diegetic UI

- Interface that is included in the game world -- i.e., it can be seen and heard by the game characters

Spatial UI

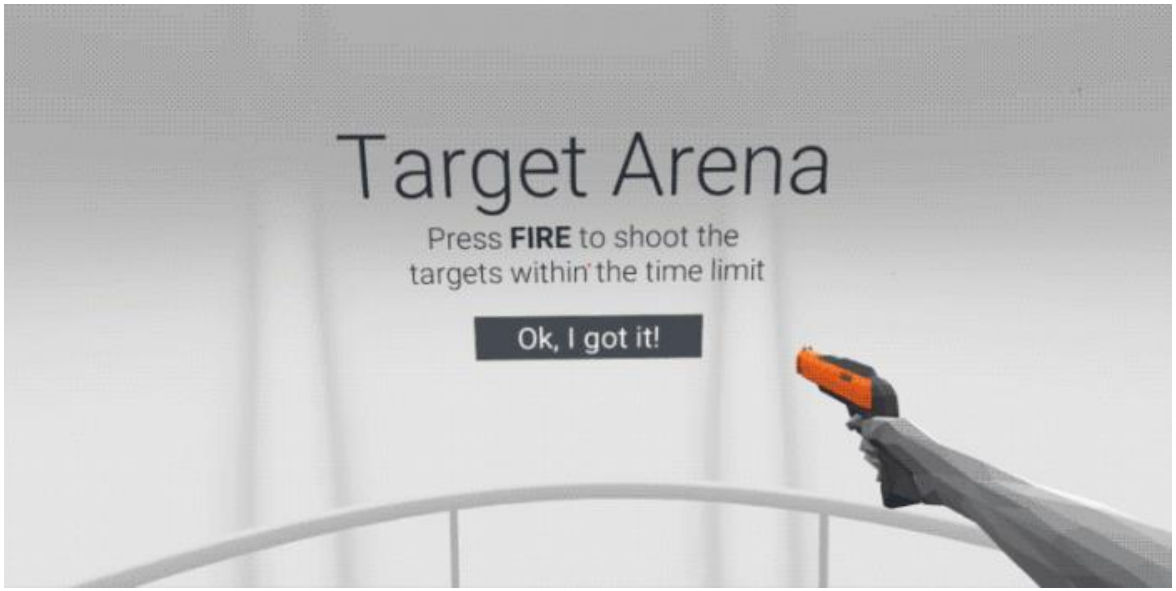
- UI inside the world
- Rendered if it were a plane object in the scene
- Plane oriented however you like





Spatial UI and Diegetic UI

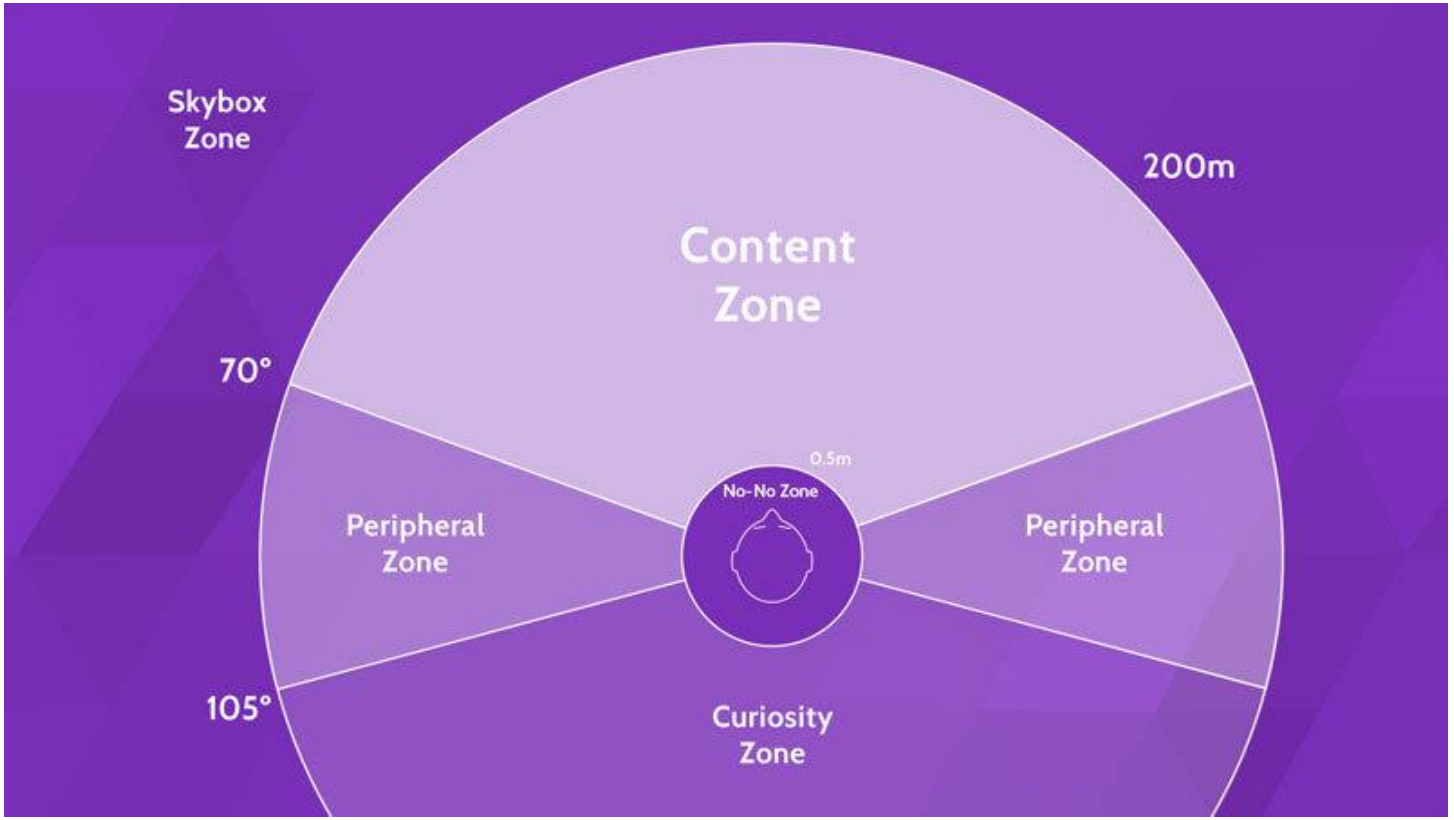
- Where have to be placed?
  - Too close to the user can cause eye strain
  - Too far away can feel like focusing on the horizon



Spatial UI and Diegetic UI

- Where have to be placed?
  - It’s best to position your UI at a comfortable reading distance and scale it accordingly.
  - Many developers will initially attach the UI to the camera, so that as the player moves around the UI will stay in a fixed position
    - This can lead to user discomfort or nausea!

Content Zones for VR

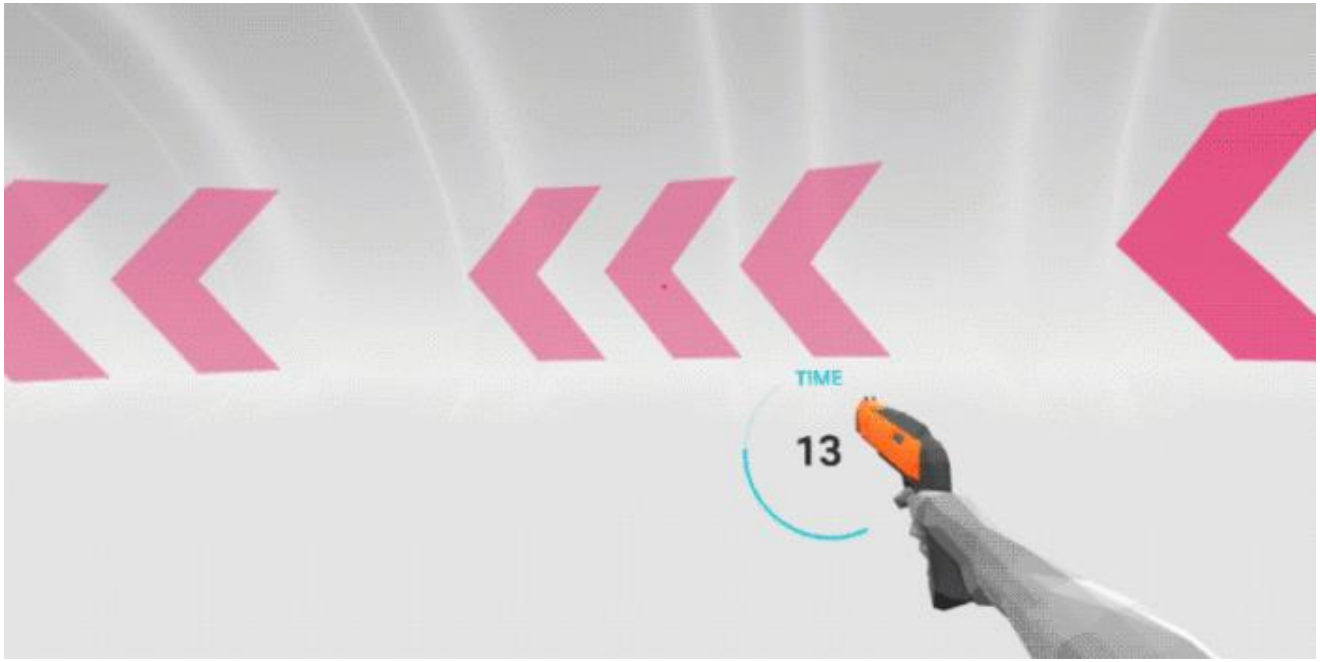


Content Zones for VR

- **Content Zone**: area of comfortable head rotation and view where things still give stereoscopic depth perception.
- **Peripheral Zone:** The area visible with maximum head rotation. Environment will still be seen more regularly, but no long-term content should be put in this zone
- **Curiosity Zone** : the user is literally turning their shoulders and trying with some effort to see what's behind them
- **No-No Zone:** As things get close to the face, the user becomes cross-eyed and experiences eye strain. Nothing should be displayed in this sphere around the head
- **Skybox Zone** - after 200m, the two displays are essentially showing the same image pixel for pixel and there is no depth perception.

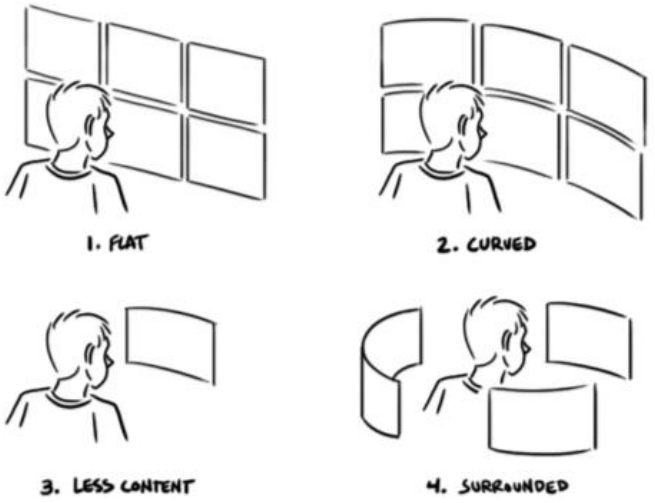
Catching user's attention

- Example of UI used to catch user's attention



Types of UI

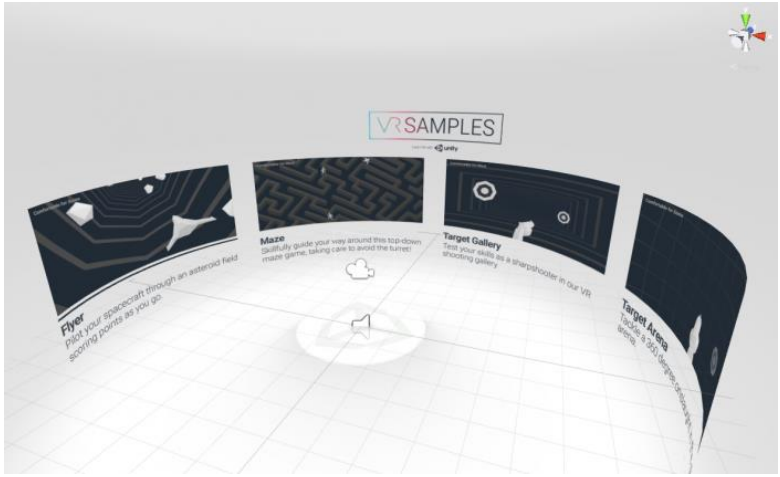
- **Flat:** difficult to read text or images in perspective. There is no sense of grounding in the space.
- **Curved**: The content is curved around the user, easier to read text or images.
- **Less content**: Better, even if that requires some way to move through it.
- **Surrounded**: Hierarchy can be implied by nearness to the cone of focus. Secondary content can be pushed out of immediate view but still remain accessible.



Curved UI in Unity

- A possible way to have curved UI in Unity is use the free package «Unity UI Extension» (it is not an official package)
  - https://bitbucket.org/UnityUIExtensions/unity-ui-extensions/src/master/

- It also has:
  - Curly UI Graphics
  - Curly UI Image
  - Curly UI Text



UI Interaction - Unity

- In VR, we frequently need to activate an object that a user is looking at.
- In Unity a solution for this problem is to use:
  - VRInput
  - VRInteractiveItem
  - VREyeRaycaster

- You can find them in VR Samples
  - https://assetstore.unity.com/packages/essentials/tutorial-projects/vr-samples-51519

VRInput

- VRInput is a simple class that determines whether swipes, taps, or double-taps have occurred.
- You can subscribe to events on *VRInput* directly:

```
public event Action OnClick; // Called when Fire1 is released and
it's not a double click.
public event Action OnDown; // Called when Fire1 is pressed.
public event Action OnUp; // Called when Fire1 is released.
public event Action OnDoubleClick; // Called when a double click
is detected.
public event Action OnCancel; // Called when Cancel is pressed.
```

VRInteractiveItem

- Component you can add to any GameObject that you would like the user to interact with.
- It requires a collider on the object it is attached to.
- There are six events you can subscribe to:
  - **OnOver, OnOut, OnClick, OnDoubleClick, OnUp, OnDown**
- And one boolean that can be used to see if it's currently Over:
  - **IsOver**

VREyeRaycaster

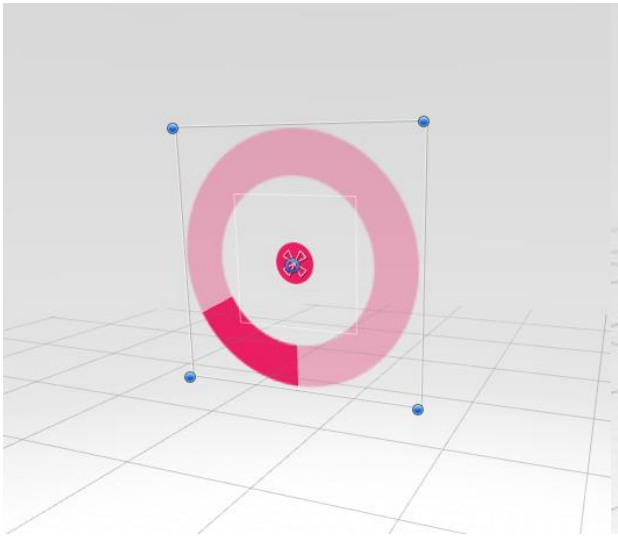
- This script needs a reference of the Main Camera.
- Every Update() the script casts a ray forwards using Physics.Raycast to see if the ray hits any colliders.

```
VRInteractiveItem interactible =
hit.collider.GetComponent<VRInteractiveItem>(); //attempt to
get the VRInteractiveItem on the hit object
```

Feedback: Everything Should Be Reactive

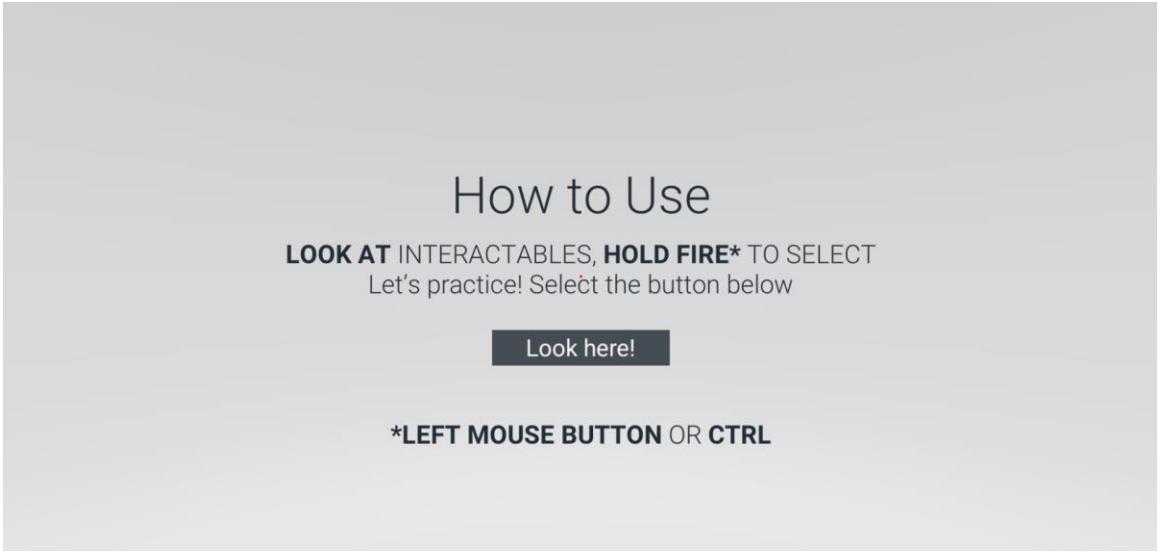
- Every interactive object should respond to any casual movement.
- Any casual touch should provoke movement
  - The kinetic response of the object coincides with a mental model, allowing people to move their muscles to interact with objects.

Interaction and feeback: SelectionRadial and SelectionSlider



Gesture Descriptions

- Text- or audio-based tutorial prompts can be essential for first-time users.
- Be as specific as possible to get the best results, using motions and gestures that track reliably.



Interactive Element Targeting

- Appropriate scaling
  - Ensures the user can accurately hit the target without accidentally triggering targets next to it.
- Limit unintended interactions
  - Be sure to space out UI elements so that users don’t accidentally trigger multiple elements.
- Limit hand interactivity
  - Make a single element of the hand able to interact with buttons and other UI elements .