COMP3421 Week 7

VR Development Basics

What do we need to consider when developing in VR?

- “Presence”, or “immersion”.
- If you stand on top of a building in a normal “on-the-screen” game, you probably won’t feel any sense of vertigo.
- This is not necessarily the case in VR. You want to try and keep in mind how the player is going to feel when they move through your environment.
- **Don’t move the camera when using a first person VR view.**
  - This means no screen shakes, no “cinematic” camera cutscenes, or anything of that nature. Anything like this is going to make your player sick. Cutting to a 3rd person view is a good solution when you want to take control of the camera. It can still be disorienting, though.
- **We’re mimicking eyes, not a camera.**
  - So also generally stay away from things like lens flares, for example.

VR Controls

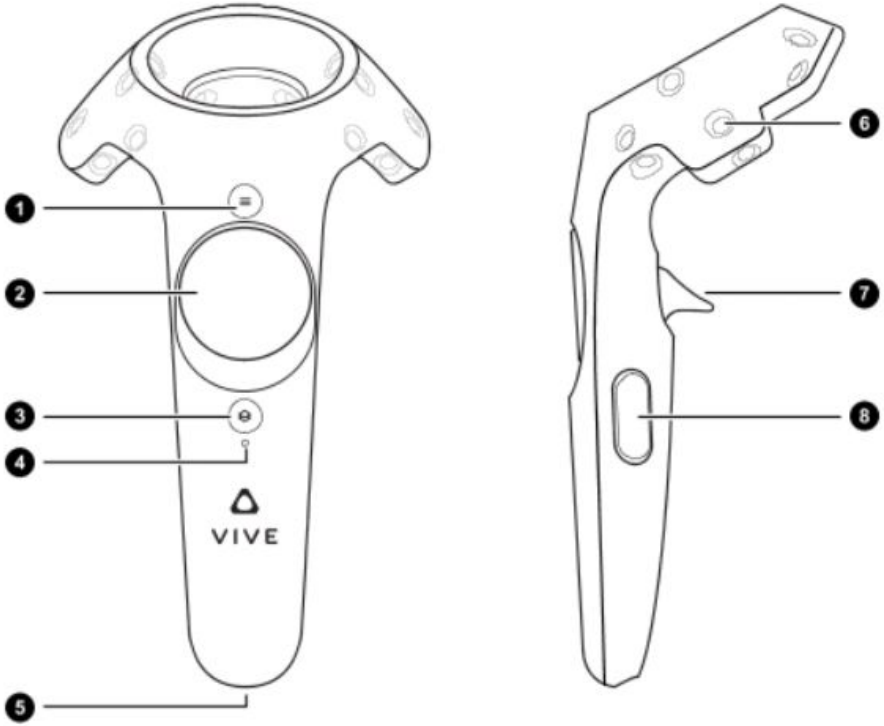
- 3DOF vs 6DOF controllers
- 3DOF controllers track rotation only
- 6DOF controllers track position as well. The HTC Vive Pro controllers fit into this category.
- VR controllers also often come with hand controllers that can also be 3DOF/6DOF.

VR Controls

Source: https://www.vive.com/us/support/vive/category_howto/about-the-controllers.html

Important:

Don't scratch the sensors!



1	Menu button
2	Trackpad
3	System button
4	Status light
5	Micro-USB port
6	Tracking sensor
7	Trigger
8	Grip button

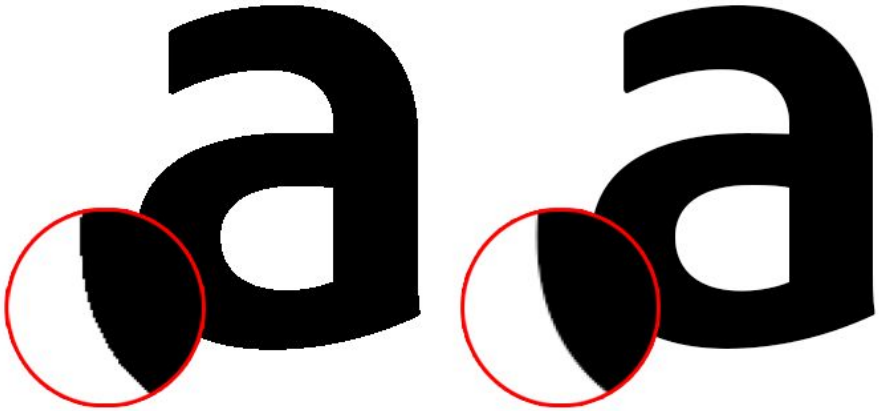
Things to consider:

- Game movement often done with trackpad
- You don’t have as many buttons as you would on a keyboard
  - May need to make actions context sensitive
  - May need some kind of menu
  - May want to bind some actions to certain movements (difficult)
- If you’re using a HUD (heads-up display), this can’t be 2D like on a screen. You’ll need to convert it to 3D somehow.
- In general, VR headsets may not have the best resolution, which might make reading lots of text tricky. Keep this in mind - it may not be too much of an issue with the Vive Pro, though.
- Again, careful when designing environments and moving the camera (be aware of how things will *feel*).

Engine Settings for VR

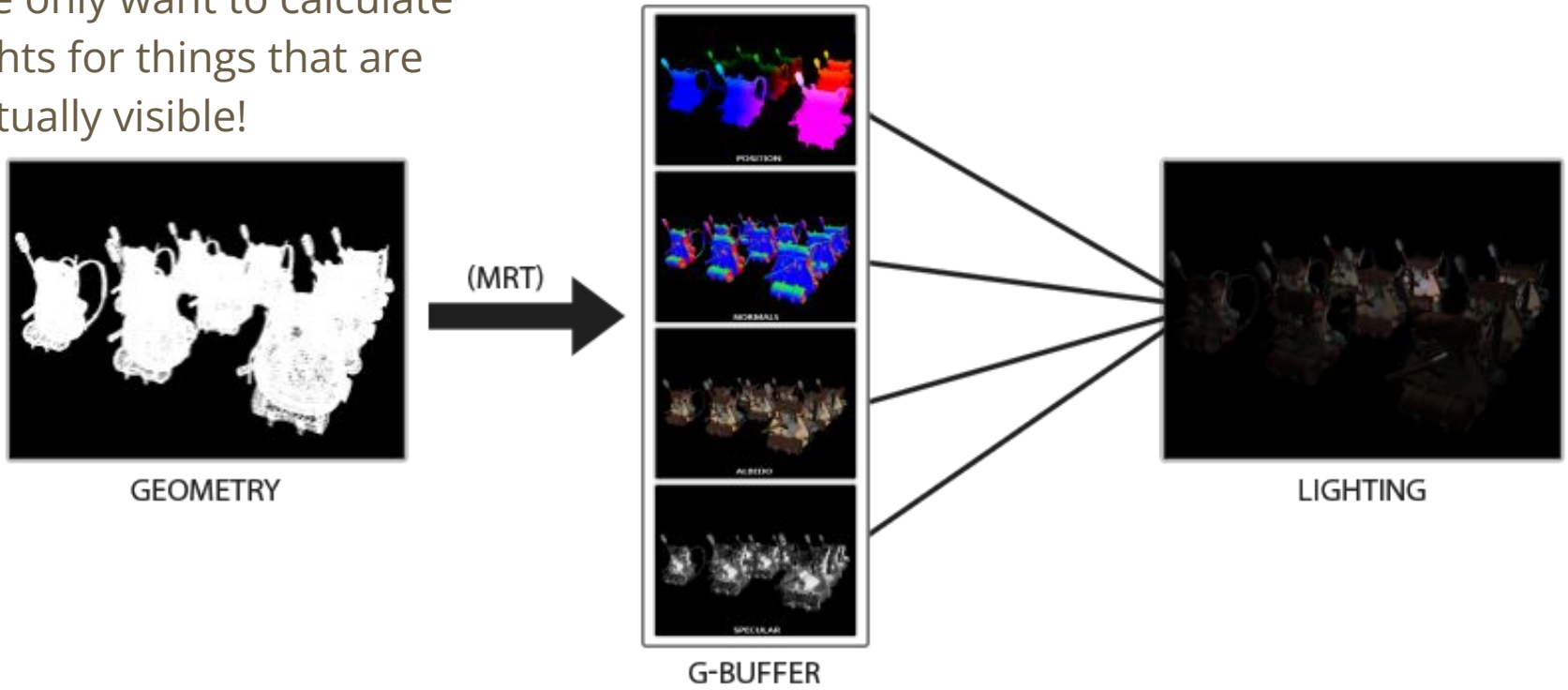
- Project Settings -> Project -> Description > Check ‘Start in VR’
- Project Settings -> Engine -> Rendering -> Check these
  - Instanced Stereo
    - Avoid rendering the scene twice (for each eye) - passes render to headset
  - Mobile HDR
    - Needed for emissive light
  - Mobile Multi-View
    - Optimised headset rendering
  - Round Robin Occlusion Queries
    - Optimises occlusion calls (which check if an actor is blocking another actors) - only does this for one eye a frame
  - Forward rendering?
    - Gives us access to MSAA (multi-sampling anti-aliasing), which works really well with VR.
- Project Settings -> Engine -> Rendering -> Anti-aliasing method -> MSAA

Anti-aliasing



Deferred rendering

- We only want to calculate lights for things that are actually visible!



However...

- Deferred rendering is great for efficiently adding many lights to our scene.
- However, we lose access to MSAA (multisample anti–aliasing) if we use it. This is one of the best options for VR...
  - https://learnopengl.com/Advanced-OpenGL/Anti-Aliasing
- Unreal has made optimisations to forward rendering such as culling lights using a frustum (shape like pyramid with the tip cut off).
- Also, as an aside, a lot of traditional screen space effects (SSR - screen space reflections, SSAO - screen space ambient occlusion), don't work particularly well in VR. Keep this in mind.
  - Screen space is the mapping of the 3D rendered space onto a 2D screen. Many effects have traditionally been done in screen space because it is relatively efficient.

Input Events/Actions in VR

- The Vive Pro has bindings for all of its buttons, but also:
- X/Y and up/down/right/left axes for the touchpad(s).
- A “grip” event (grip button).
  - Can also be used as an axis for things like animation - see the week 7 Monday lecture.
- A “trigger” event (trigger press)
- A “trigger” axis (i.e. how far down are you holding the trigger button?)
- A “trackpad touch” event
- Some more...
- We use motion controller components in the VR player class to track the position of our hand controllers