CS 498vr

Virtual Reality
UI and Movement methodology
(plus a recap of Design/Production)

CS 498 VR Design Elements

Design

- Describe key elements of design required for effective VR development
  - Tools for developing your design
    - High Level Vision Canvas
    - Action Loops / Beat by Beat
    - Feature List
  - General Design processes/considerations
  - Ergonomics
  - Interface
  - Immersion
  - Movement techniques

High Level Vision Canvas

Project Name

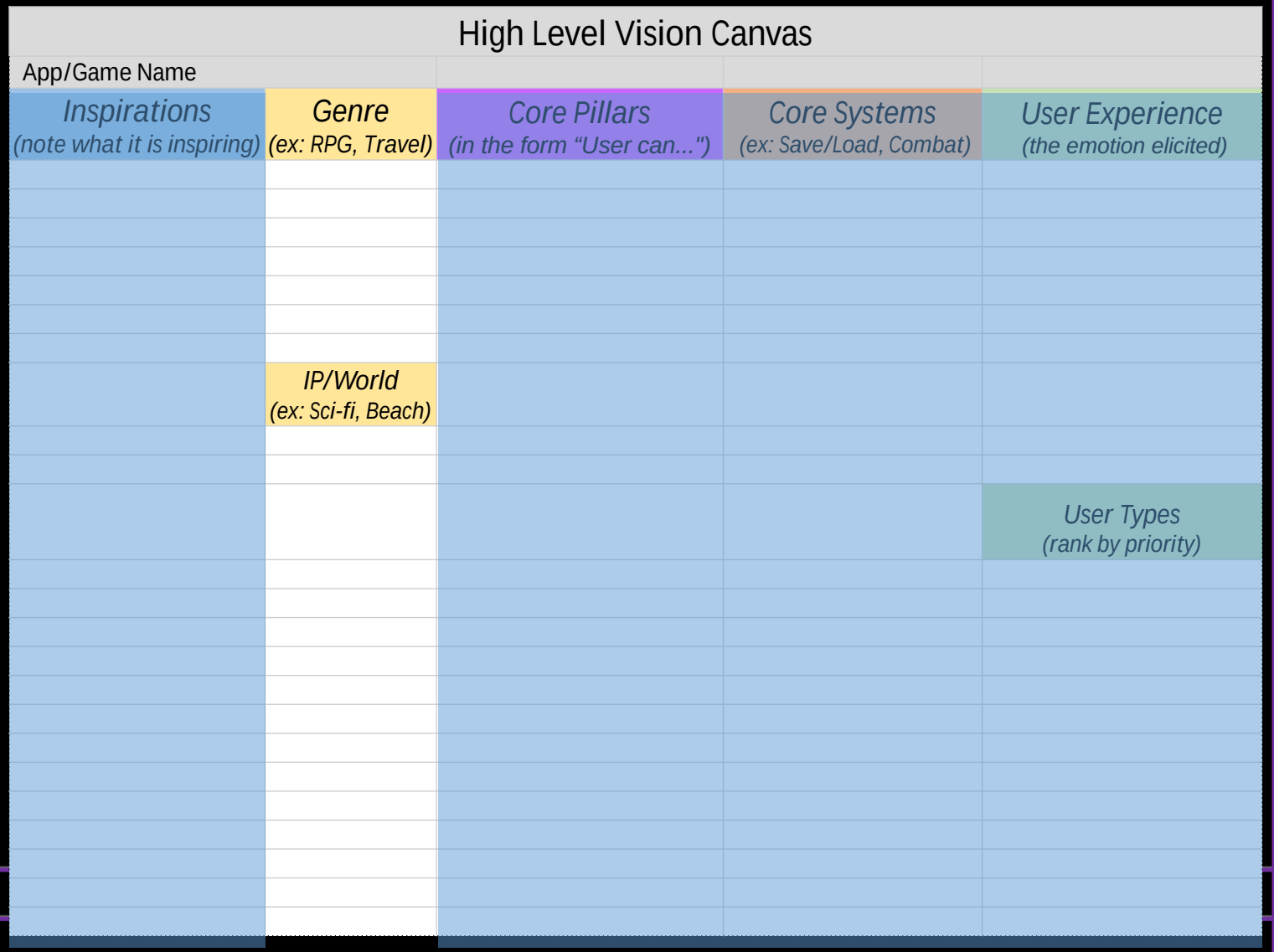
Inspirations *(note what is it inspiring)*	*Genre* *(ex: 3rd Person)*	*Core Pillars* *(in the form "User can...")*	*Core Systems* *(ex: Save/Load or Combat)*	*User Experience* *(the emotion elicited)*
Inspirations inform about what the usage, gameplay, tone (humor or atmosphere), UI, Audio, and art style are like by providing similar examples	**The Genre describes, informs, and constrains the usage and camera style and sets expectations for the team what the team will be building for the User. These should be common genres**	**The core pillars are the actions the user can perform that define the user actions (the core mechanics) that, if removed would change the user experience.**	**Core Systems are the key programmatic features/systems that need to be created to build your project, they should directly related to Core Pillars and should also contain those systems needed that are not core but necessary (for example Save/Load)**	**The User experiences should be the foundation for your core mechanics (Core Pillars and Core Systems). They should be the emotional experiences you are trying to evoke from the user.**
Inspirations are key, they show others what surrounds your idea and makes it special.	Don't put world desciptors here (Zombies are not a Genre, they are an element of the world, the genre would be Survival)	Make sure each core pillar reflects/supports a user experience.	Don't forget those background elements that drive key game play mechanics (AI, Audio, UI, etc.) These are key systems.	If you want to build something that has an impression on users, then you need to develop this element properly, with real consideration for experiences you have had as well as emotions you want to drive.
Remember to note what part of the game each element inspires.		Grouping like elements or sub elements is good for clarity (ex: pillars about ranged and melee combat would fit under a more general combat pillar)	The details in this and core pillars drive the feature list. Be thoughtful here, start with higher level elements and then add contributing background elements).	If you don't have much to offer here you may need to consider adding depth to what you are building (if a tour say, you might want to gamify it in some way).
	IP/World *(high level descriptions)*	Don't put emotions here, they belong in User Experiences. They can be rewritten as core pillars.		*User Types (prioritized)*
	The IP/World element is there to provide context for where the game/project is being placed. Elements about Narrative and environment belong here.			**These are the types of users you think your project will attract. These should be prioritized by size of group (biggest first). This list helps you make decisions when conflicts arise between supporting features and handling scope.**
	This is more important that most designers realize. Adding enough detail to describe the context of what you are building in just a few lines of text is key.			We found that personalizing the Names for these groups helped describe them. Knowing these groups well helps define your efforts.
	Timeframes matter (For example: Sci fi Technology can vary greatly based on how far in the future we are talking about.			You don't have to be one of these types but you need to find these types to try your app to make sure it is meeting their needs.

Vision Canvas – Single Project/Level Version

Determine the Descriptors

- Define the Genre
  - 1st Person or 3rd, Travelogue, Shooter
  - Don't include info about the world
  - Expectation is for standard Genre vernacular
- Describe the World/Environment or IP
  - Information about the background of the world
    - Post Apocalyptic, Sci Fi, Japan, Beach , etc.
  - Licensed Intellectual Properties, etc.
  - Concepts
  - Narrative

These define the physical world, supplying context, interactions and connections. They describe how the user interacts (ex: camera view) and relates (especially for a existing IP).



Genre *(ex: 3rd Person)*
The Genre describes, informs, and constrains the usage and camera style and sets expectations for the team what the team will be building for the User. These should be common genres
Don't put world desciptors here (Zombies are not a Genre, they are an element of the world, the genre would be Survival)
IP/World *(high level descriptions)*
The IP/World element is there to provide context for where the game/project is being placed. **Elements about Narrative and environment belong here.**
This is more important that most designers realize. Adding enough detail to describe the context of what you are building in just a few lines of text is key.
Timeframes matter (For example: Sci-fi tech. can vary greatly based on how far in the future we are talking about).

Vision Canvas – Single Project/Level Version

Descriptors, continued:

Write down all your inspirations

- Get specific
  - UI
  - Game play features
  - Tone (Tone Targets)
  - Style (Graphic look)
  - Lighting

This helps define the emotional elements the user will be experiencing.

Inspirations *(note what is it inspiring)*
Inspirations inform about what the usage/gameplay, tone (humor or atmosphere), UI, Audio, and art style are like by providing similar examples.
Inspirations are key, they show others what surrounds your idea and makes it special. Remember to note what part of the game each element inspires.

Vision Canvas – Single Project/Level Version

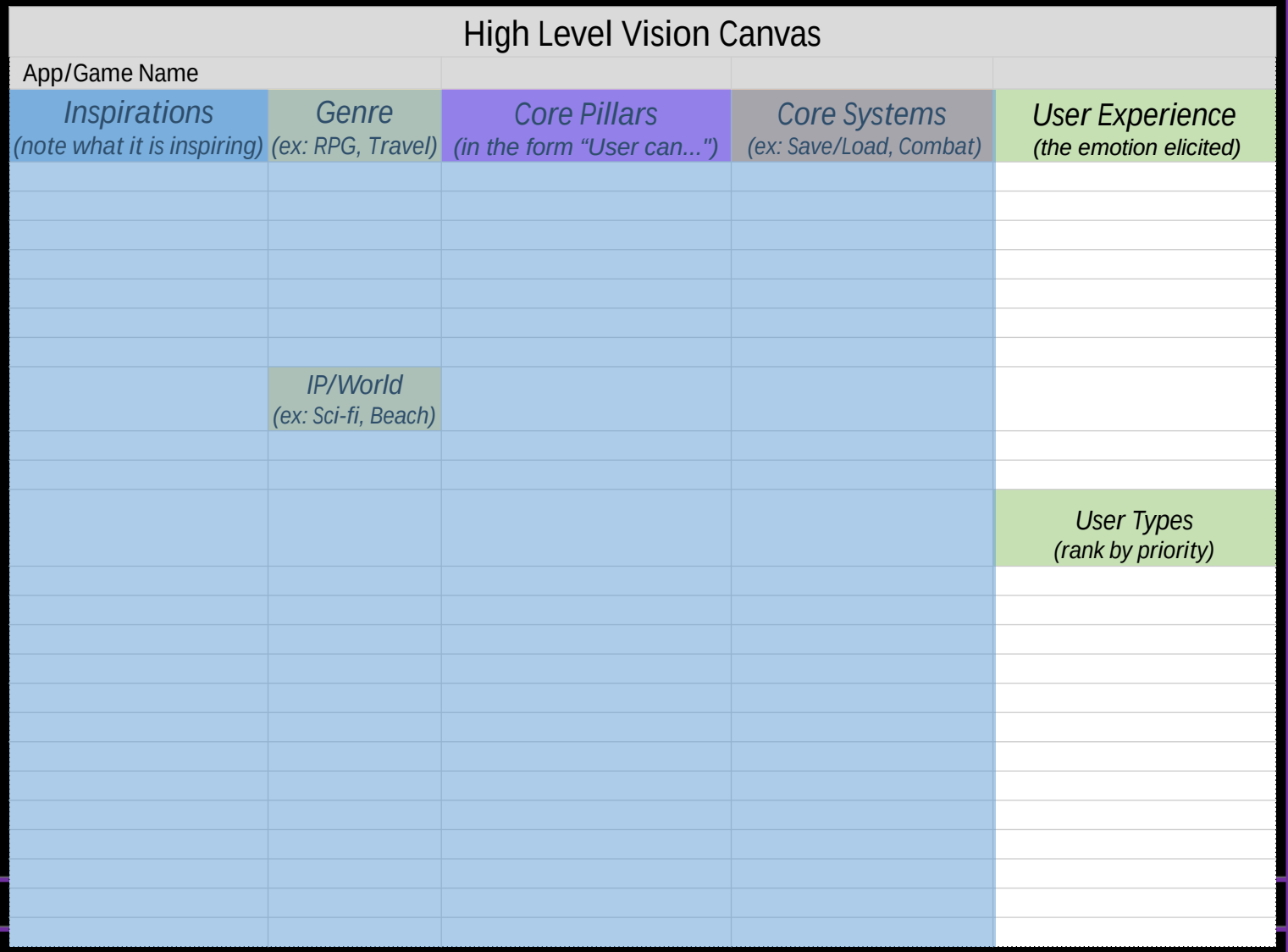
Target Information

User Experience

- The emotional experience the user is having
- Should relate to core pillars

Player Types

- Self defined
- Should be descriptive
- Should represent the key player/purchasers
  - In descending order
- Don't go too broad
- Make up your own descriptors



User Experience (the emotion elicited)
The User experiences should be the foundation for your core mechanics (Core Pillars and Core Systems). They should be the emotional experiences you are trying to evoke from the user.
If you want to build something that has an impression on users, then you need to develop this element properly, with real consideration for experiences you have had as well as emotions you want to drive.
If you don't have much to offer here you may need to consider adding depth to what you are building (if it's a tour say, you might want to gamify it in some way).
User Types (prioritized)
These are the types of users you think your project will attract. These should be prioritized by size of group (biggest first). This list helps you make decisions when conflicts arise between supporting features and handling scope.
We found that personalizing the Names for these groups helped describe them. Knowing these groups well helps define your efforts.
You don't have to be one of these types but you need to find these types to try your app to make sure it is meeting their needs.

Vision Canvas – Single Project/Level Version

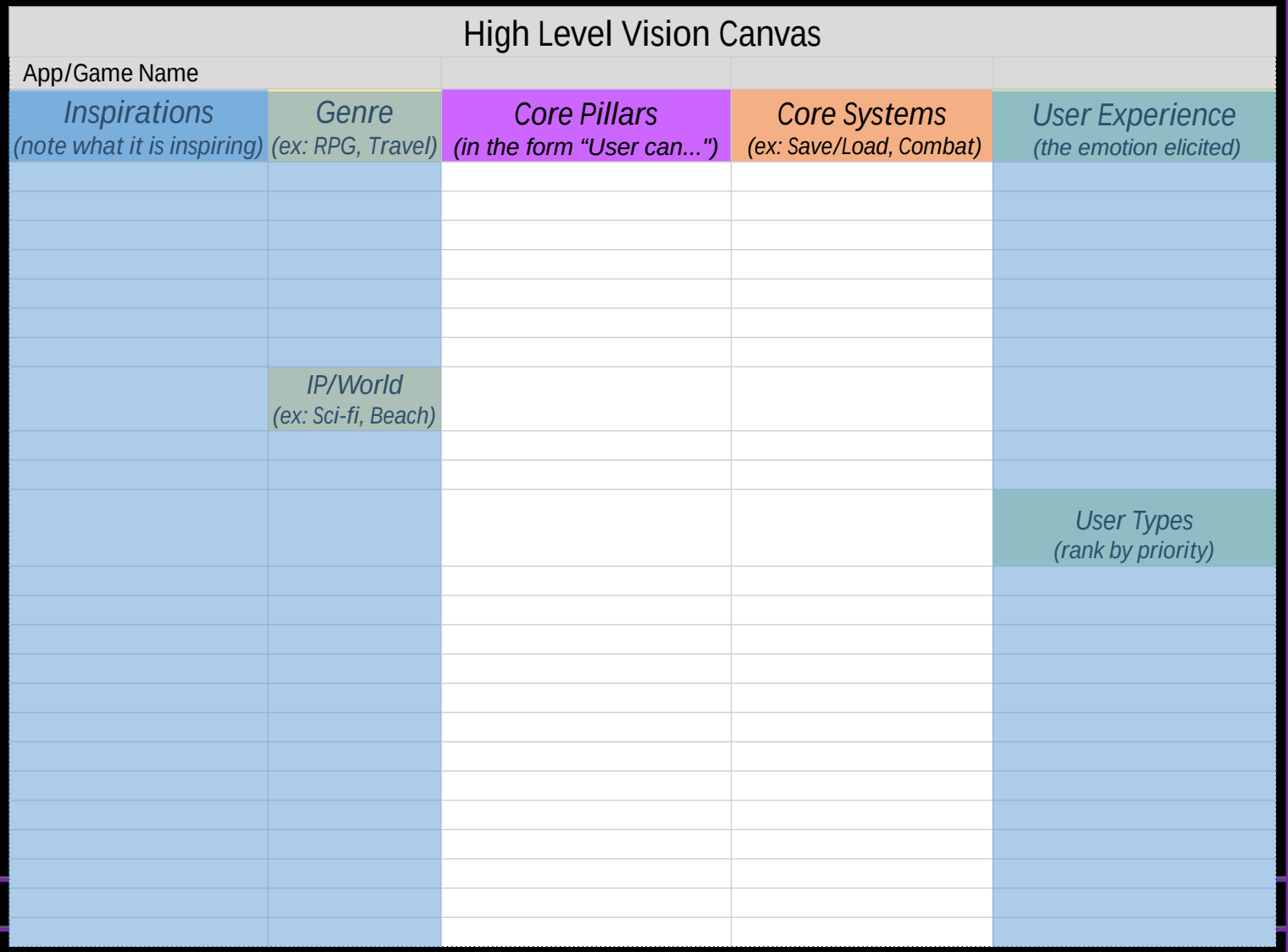
Foundational Elements

Core Pillars

- Features that drive the User Experience
- In the form: *"The User will ....."*

Core Gameplay Systems

- Systems that support the core pillars and other key aspects of the app/game
- Points to key low level systems
- Examples: Traversal, UI, AI (if needed).



Core Pillars *(in the form "User can...")*
The core pillars are the actions the user can perform that define the user actions (the core mechanics) that, if removed would change the user experience.
Make sure each core pillar reflects/supports a user experience.
Grouping like elements or sub elements is good for clarity (ex: pillars about ranged and melee combat would fit under a more general combat pillar)
Don’t put emotions here, they belong in User Experiences. They can be rewritten as core pillars.

Core Systems *(ex: Save/Load or Combat)*
Core Systems are the key programmatic features/systems that need to be created to build your project, they should directly related to Core Pillars and should also contain those systems needed that are not core but necessary (for example Save/Load)
Don't forget those background elements that drive key game play mechanics (AI, Audio, UI, etc.) These are key systems.
The details in this and core pillars drive the feature list. Be thoughtful here, start with higher level elements and then add contributing background elements).

A Simple Project Plan

Contains:

- High and mid-level Design Elements
  - Vision Canvas
  - Feature List
  - Action Loops
- Simple Technical Design Doc
- Simple Milestone doc (instead of a true schedule)
- QA test plan

Simple Feature List

Number	Feature	Top Level Description	Priority	Difficulty	Risk/Considerations	Comments
F001	Melee Combat	Primary combat system	High	High	Animations may require modifications to the animation system	
					Camera system needs to pan relative to the attack type	
					VFX needs to differ for each weapon (may impact schedule)	
F002	Quadripeds	We want to support enemies with 4 legs	High	High	The volume of animations	We may be able to use the regular system and fake it
F003	Clouds	Add moving clouds to the sky dome	Low	Moderate	New system	Nice to have if possible
					Players may expect weather	
F004	Cover	Ability to hide behind objects	Med	Med	Objects need to be of certain heights, specs need to be created	Would make the combat system more interesting
					May add complexity to the combat swings	

Technical Design Document: Built from the Feature List

Breaks down the Feature List into technical appropriate segments

Specifics will be programming elements

- Systems (combinations of related tasks/features)
- One-off features

Clarifies the priorities, risks and dependencies

These are the tasks that will populate your schedule so it needs to be detailed and comprehensive.

Risk is mainly about whether something is "new". For example, a large system that you had written before would not be as risky as a small system that has been done before but is key to many other parts of the game (dependencies).

The goal is to find out which items need to be at the front of the schedule due to priority, dependency or risk.

Ver 1.1	Modified 9/11/2018							
Ref #	System Name	Feature Supported	Feature Specifics	Priority	Difficulty	Risk	Time Estimation	Comments
F001	Traversal							
F001a		Camera		High	Med	Low		
			View Frustum					To Be determined
			Viewing Distance (Scale)					Based on Frustrum
F001b		Controls		Med	Low	Low		Timing based on Environmental layout
			Walk					
			Run					
			Jump					
			Crouch					
F001c		Determine Player Location		Low	Low	Low		
F002	Melee Combat							
F002a		Camera		High	Med	Med		
			Over the Shoulder					
F002b		Controls		High	Med	Low		
			Thrust					
			Swing					
			Counter					
			Block					

Milestone Plan

Weekly or Bi-weekly set of tasks to be done by the team, tracked to:

- Show progress
- Expose scope issues
- Determine risks
- Ensure that the correct dependencies are being considered

- Milestones are team driven in this example. What will the team be accomplishing in each of the milestones over the course of the project.
- Each milestone should be reviewed and time taken to determine if the next milestone should be modified. Milestones change all the time due to new tasks, required modifications of previously completed tasks and delayed tasks.
- Be diligent here, review and rework. It is key to understanding the scope and deciding what if anything needs to change in your project based on looming deadlines.
- If new task pop up they should be put on the Feature list and be pushed down through the task list to the milestone schedule. This ensure that they get the right priority and the p;rocess of task determination (in the TDD) is correct.

Project:	Your Project name goes here				
Date	Milestone	Task	Assets Needed	% Complete	Comments
1/30/19	Design				
		Create Vision Canvas	None		
		Create Tone Target	None		
		Create Art Spectrum	None		
2/15/19	Design II				
		Create Tech Design Doc	None		
		Create Feature List	None		
	Feature I				
		Ex: Inital Play Area layout	Example: Initial UI elements		
			Initial Contact Interface		
			Initial Movement System, etc		
Insert New Dates	Insert new Milestones	Insert New Tasks	Insert New Asset Lists		Add new rows to include new milestones, add as many as you need

Quality Assurance (QA) Test Plan

Details out the elements of each feature to ensure it is working properly

- Once you reach Alpha the test plan becomes key to a successful User Experience
- Have someone not on the team, run through the test plan
- Remember to test integration, where system interact and work together
    - Most often failures occur here

There shiould be an element for every TDD item as well as items that are higher level (integration of the items on the TDD, general functionality, flow (passing from one routine to another), etc.

QA Test Plan		Project:_____	
Date	Build #	Tester Name	
1/14/2019			
Test #	Unit/Feature	Item to be Tested	Success/Fail
1	*Main Screen*		
1a		Game Loads properly to Main Screen	
1b		UI elements change focus correctly	
1c		Choosing <Start Game> functions correctly by placing the player in the game at the beginning of the first element	
1d		Choosing <Exit Game> functions correctly closing out the game cleanly and places the player back in the platform interface	
1e		Choosing <Load Saved Game> functions correctly takes the player to the <Save/Load Game system	
1f		Choosing <Settings> functions correctly by taking you to the Player Setting System	
2	*Performance*		
2a		Game loading time is within spec (loads within XX seconds)	
2b		General game play shows no hitching or graphic errors	
3	*Save Game*		
3a		Game save time is within spec (saves within XX seconds)	
3b		Saved Game can be reloaded immediately	
3c		Saved Game Data is correct on reloading	

VR Design Specifics: UI: Text in VR

Patrick O'Luanaigh (nDreams), Mike Alger, (Google)

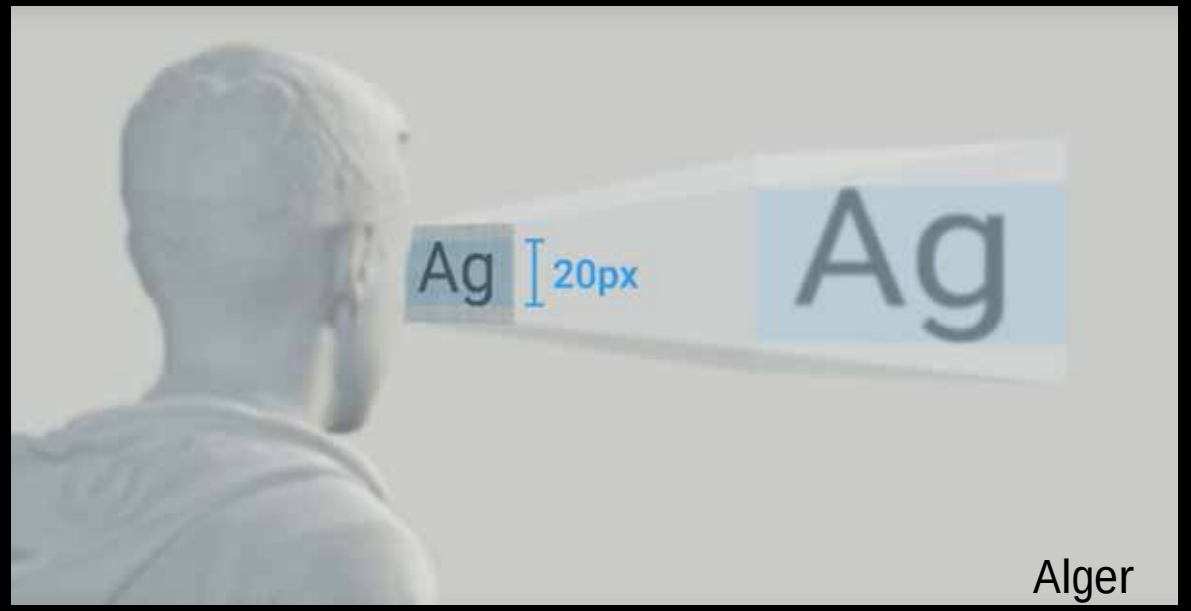
User presence as an issue related to informing the player

- Textual interfaces are an intrusion
  - Text will never completely go away
  - Put it only where it's really needed

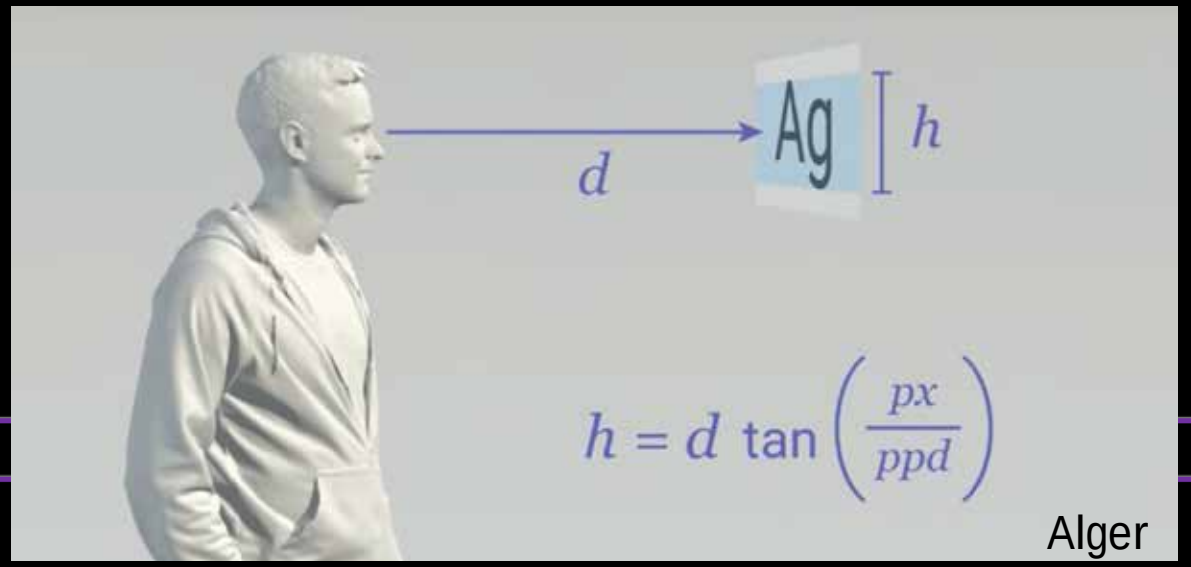
Resolution is a major constraint

- Leads to aliasing with hard edges
  - Anti-aliasing helps but impacts performance

  Avoid hard edges and thin lines
- Text Resolution is impacted
  - Larger is better
  - Avoid showing text on white and translucent backgrounds.
  - Current HMDs are about 13 PPD (pixels per degree)
    - Text should be ~1.5° degrees tall
    - Or roughly 20px tall on most current displays
  - It should be >1.3 meters from the user



Alger



Alger

VR Design Specifics: UI, where does it belong

Patrick O'Luanaigh (nDreams)

Minimal is best

Diegetic methods work best for VR

- User presence as an issue related to informing the player
  - Textual interfaces are an intrusion
    - Text will never completely go away
    - Put it only where it’s really needed
- If you need interface in the Content Area
  - Consider the Peripheral Zone
- Better would be an on-arm or on controller interface
- Gaze interface is interested

VR Design Specifics: UI, where does it belong

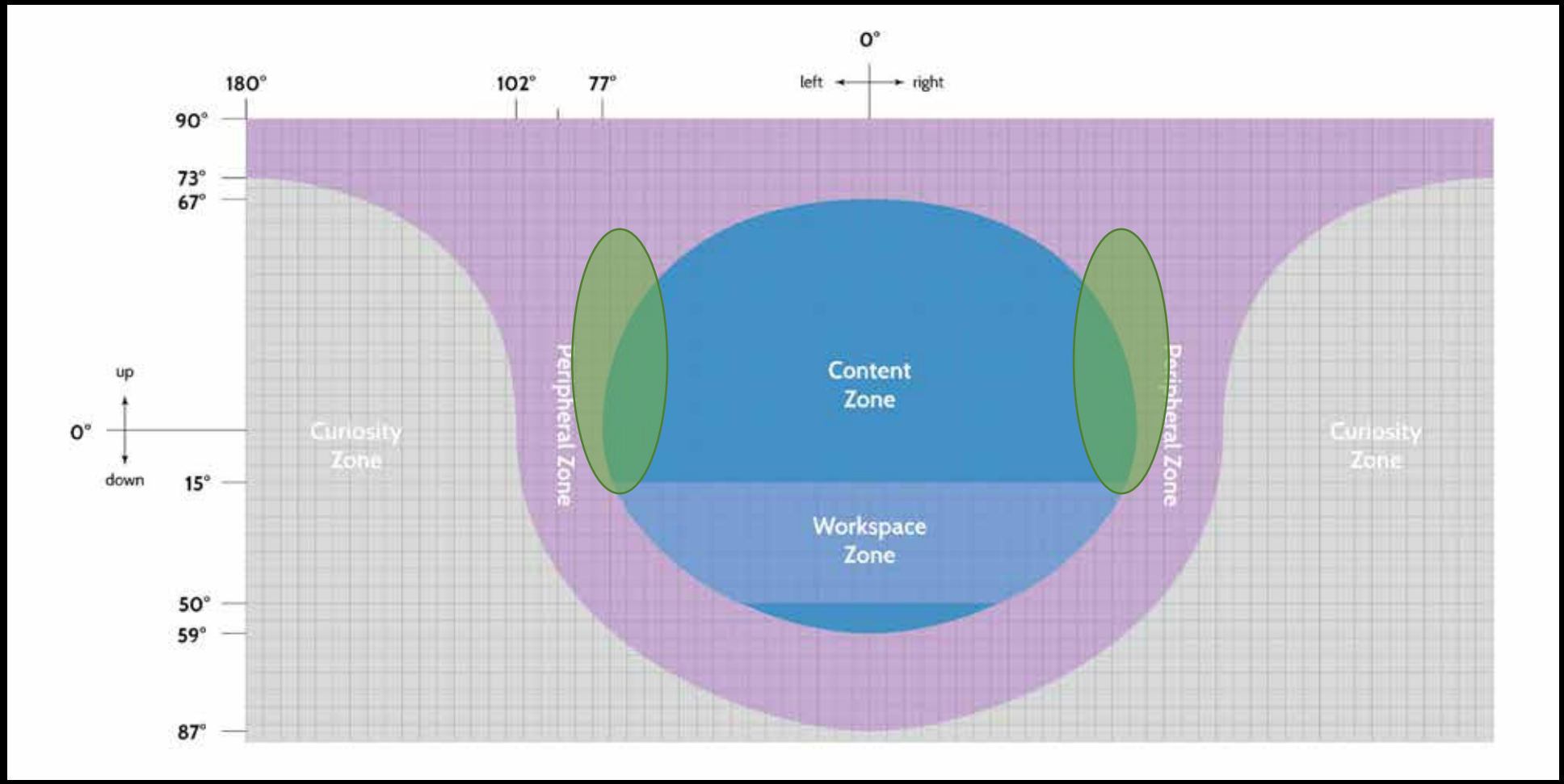
Patrick O'Luanaigh (nDreams)

Minimal is best

Diegetic methods work best for VR

- If you need interface in the Content Area
  - Consider the Peripheral Zone
- Better would be an on-arm or on controller interface
- Gaze based interface is interested

VR Design Specifics: UI, examples: UI in the content zone (*Mike Alger*)



VR Design Specifics: UI, examples: UI in the periphery of the content zone (*Dino Frontier*)

VR Design Specifics: UI, examples on Controller

(*xrdesign*)

VR Design Specifics: UI, examples on hand/arm

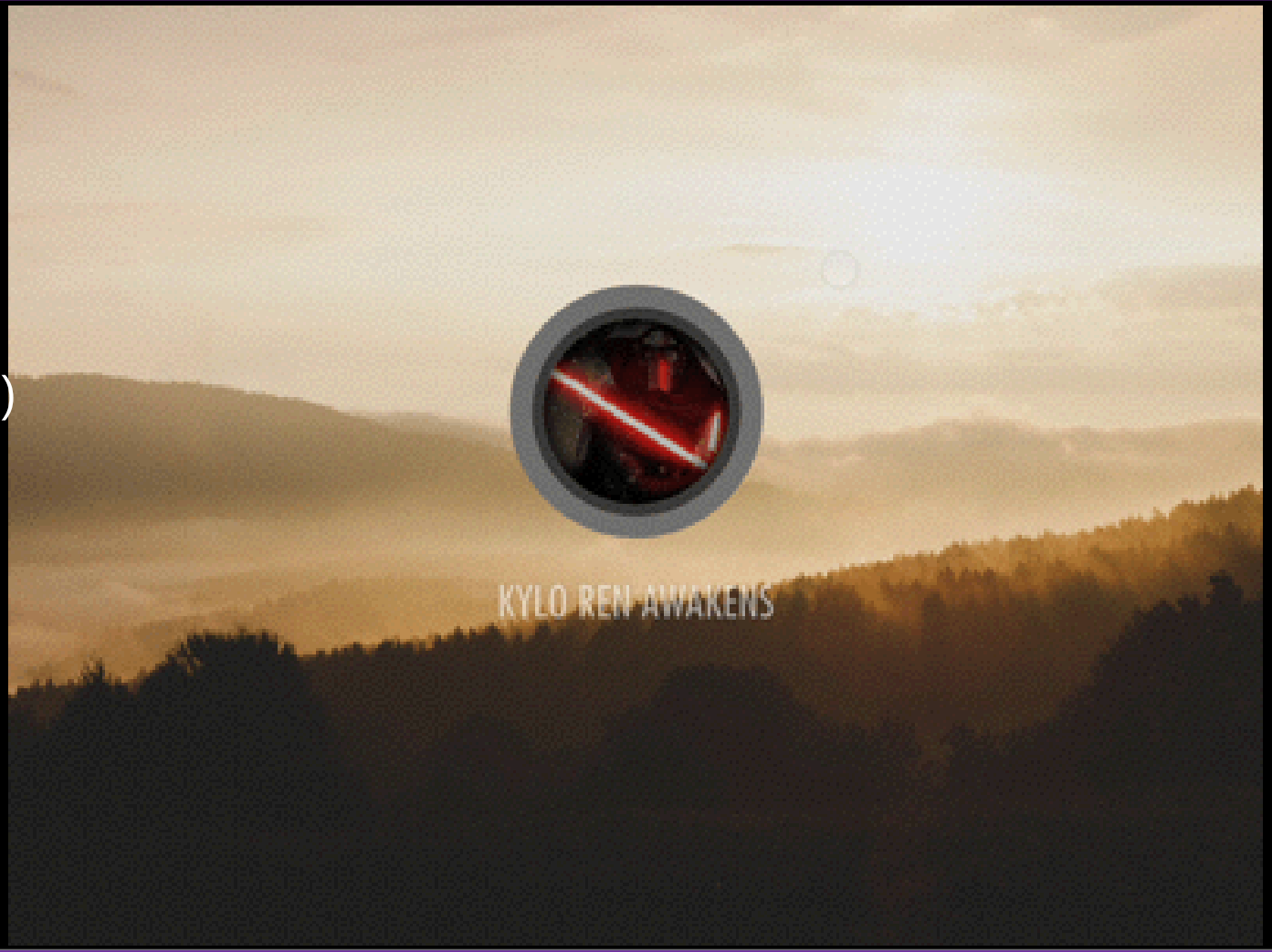
(Leapmotion hand tracking)

VR Design Specifics: UI, examples: Gaze Based
(Disney)

We can track the head position

Ray cast and provide feedback as to cursor position

Remember user gaze often returns to the center
- Move follow on menu choices off center
- Feedback, for everything (hover, choose, chosen)
  - Cursor flash
  - Color change
  - Menu item color change (selected)

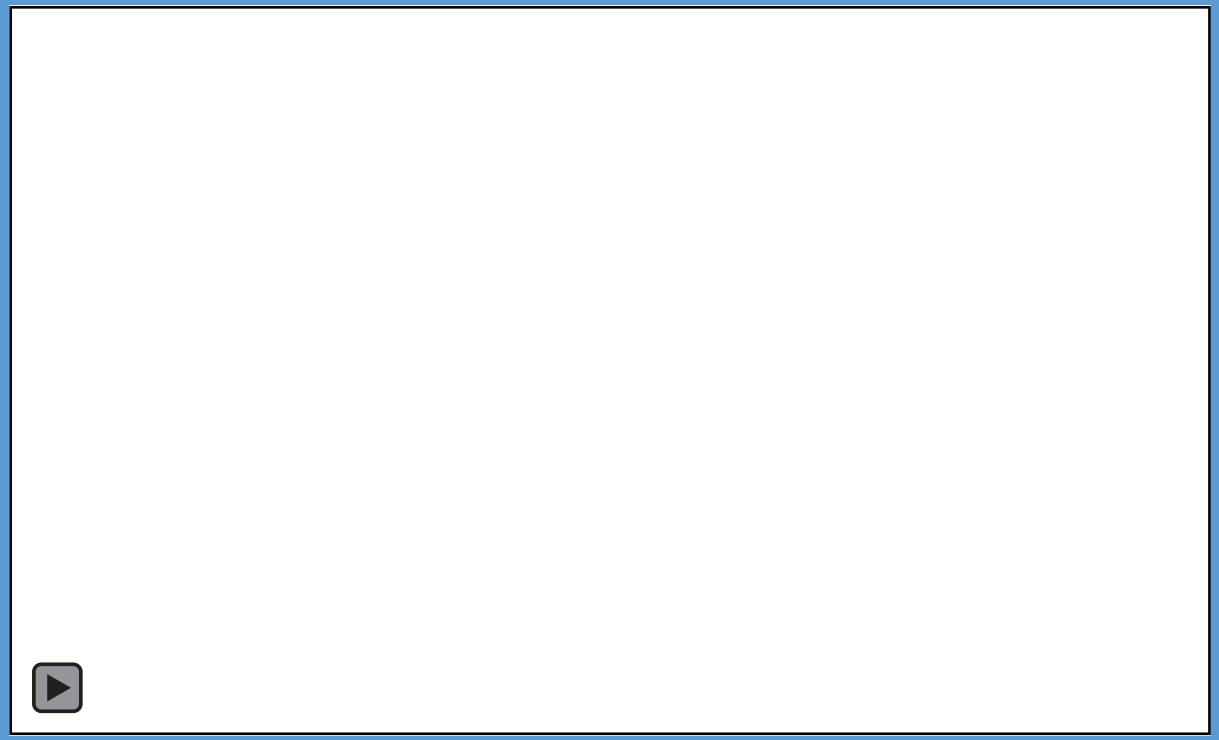


VR Design Specifics: Interesting UI examples and widgets

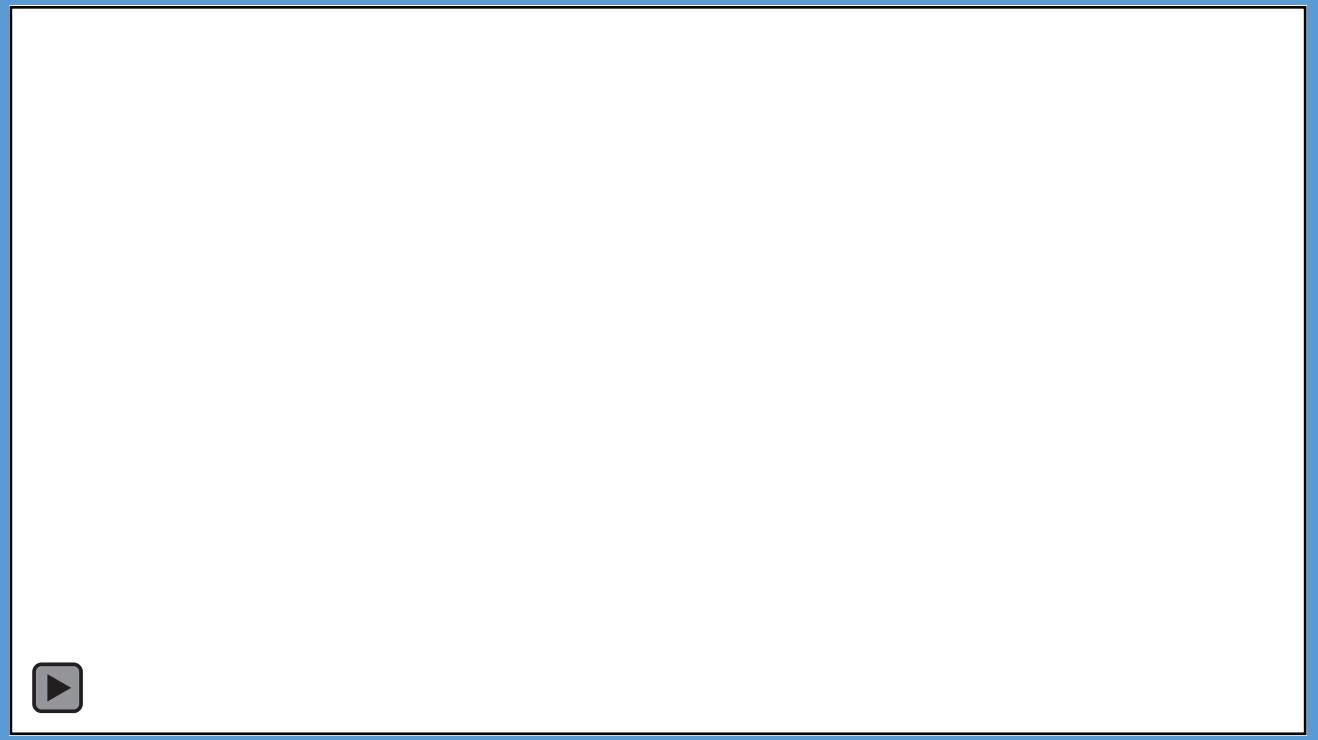
Oculus Menu example

VR Design Specifics: Interesting UI examples and widgets

Load by Face The Lab (Valve)



Load from Back



VR Design Specifics: Interesting UI examples and widgets

Tube Menu

VR Design Specifics

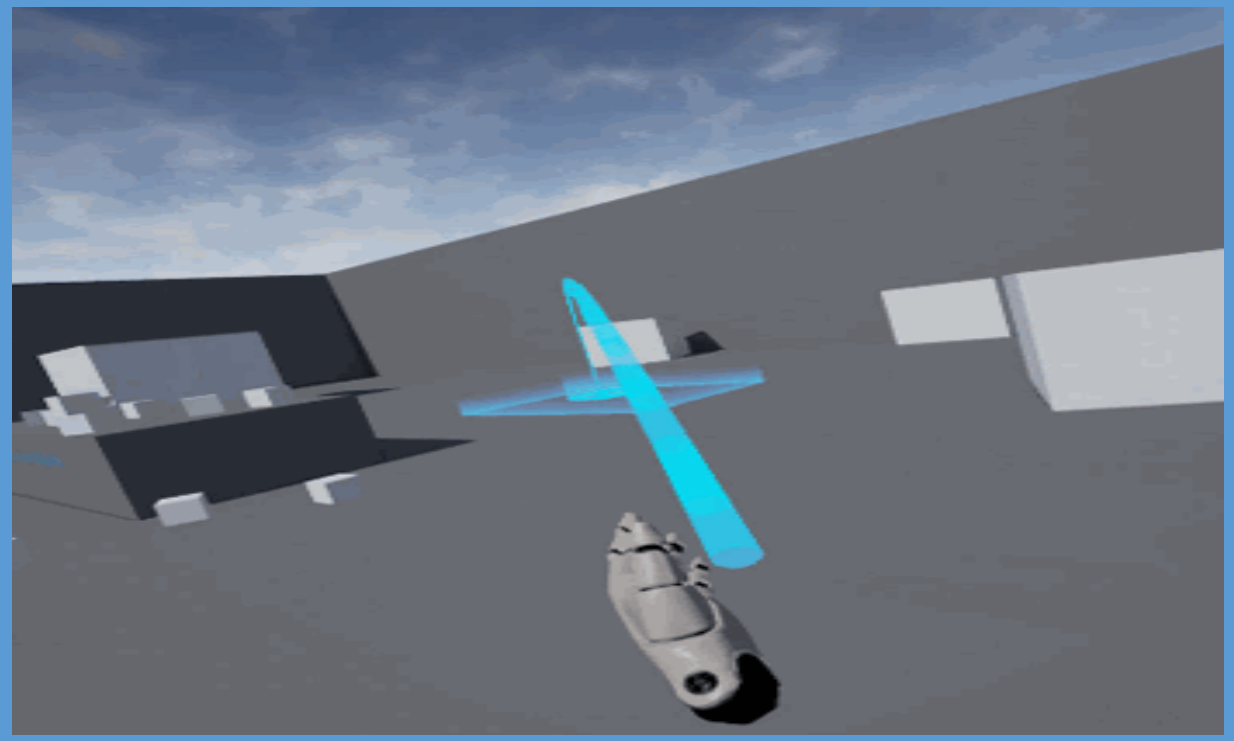
Movement Methods:

- Generally movement shouldn’t be at the speed humans normally walk
  - Faster or slower is better
  - No acceleration
  - Don’t turn
- Teleportation
  - Avoid changing orientation
  - Avoid acceleration while teleporting
  - Use frames of reference to anchor the destination (appropriate scale changes to objects)
- Tunneling
  - Masking/removing the peripheral elements can help

https://www.youtube.com/watch?time_continue=31&v=9fgkpM6211w

VR Design Specifics: Teleportation movement

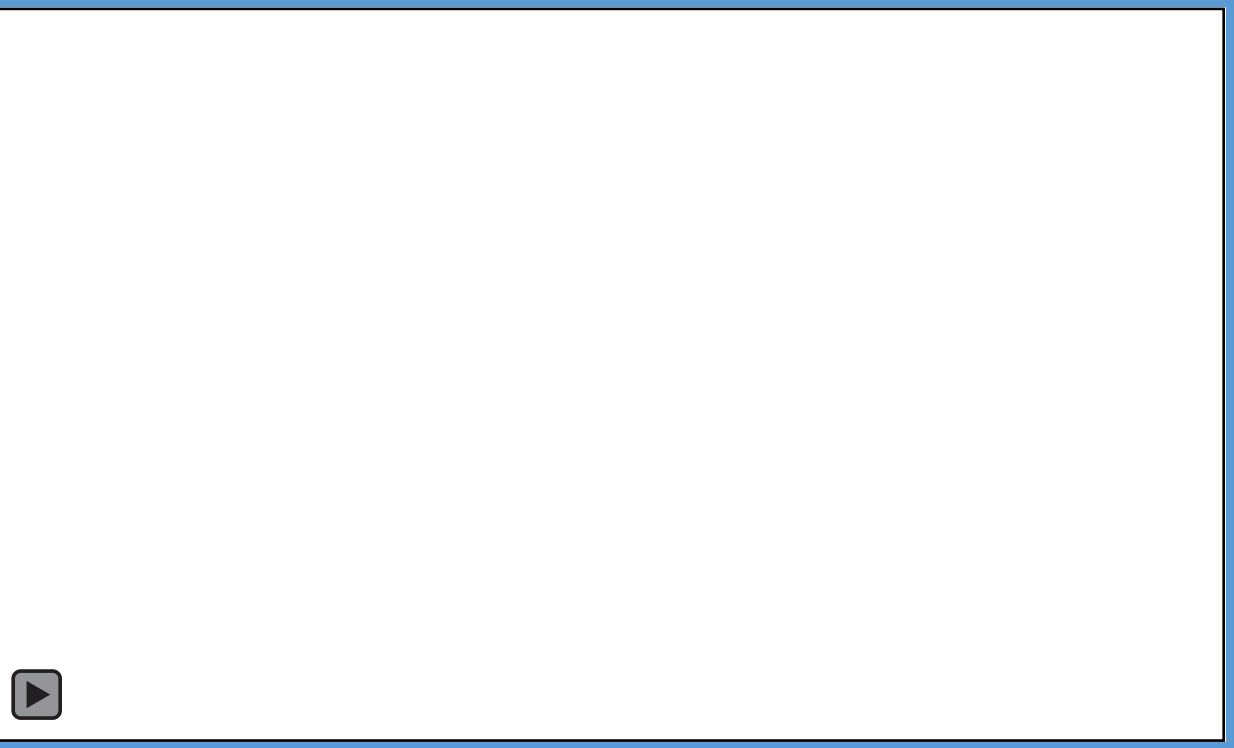
Guided Teleport *(Raw Data)*



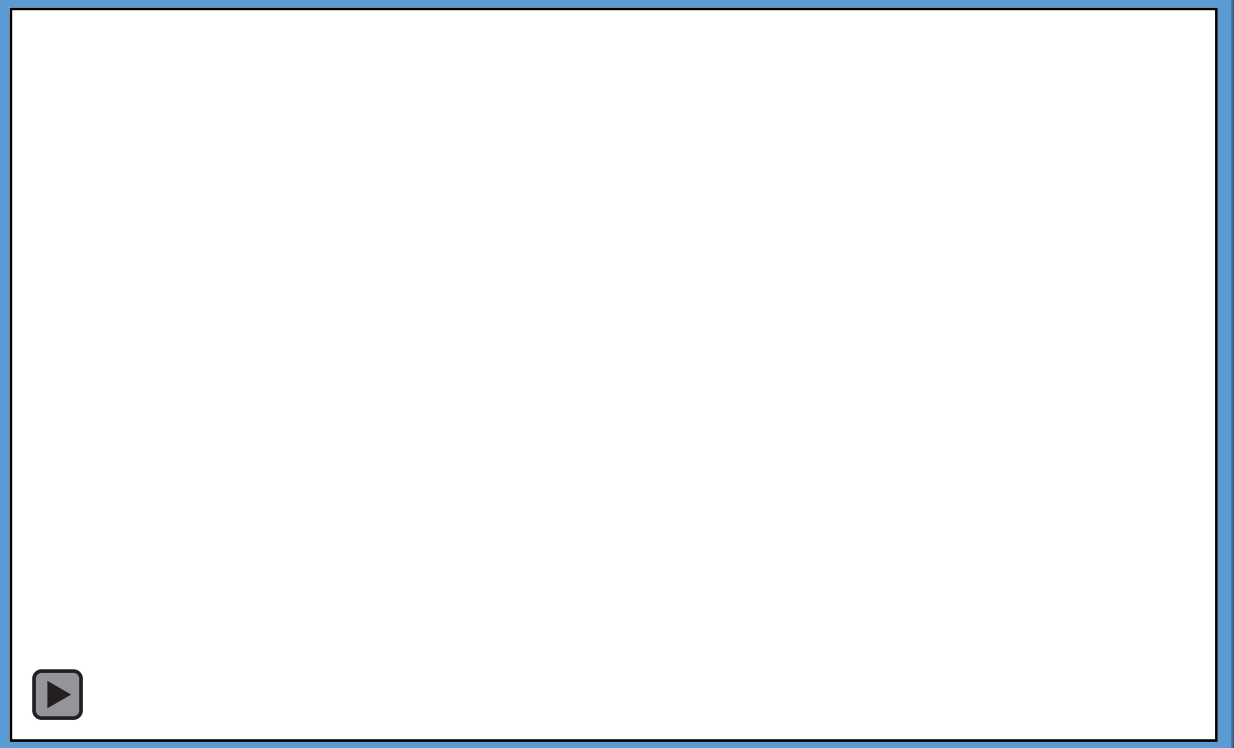
Restricted Guided *Teleport (Valve)*

VR Design Specifics: Teleportation movement

Dash *(Raw Data)*



Gaze Teleport *(vrDirect)*



VR Design Specifics: Turning movement

Drag Turning

Snap Turn

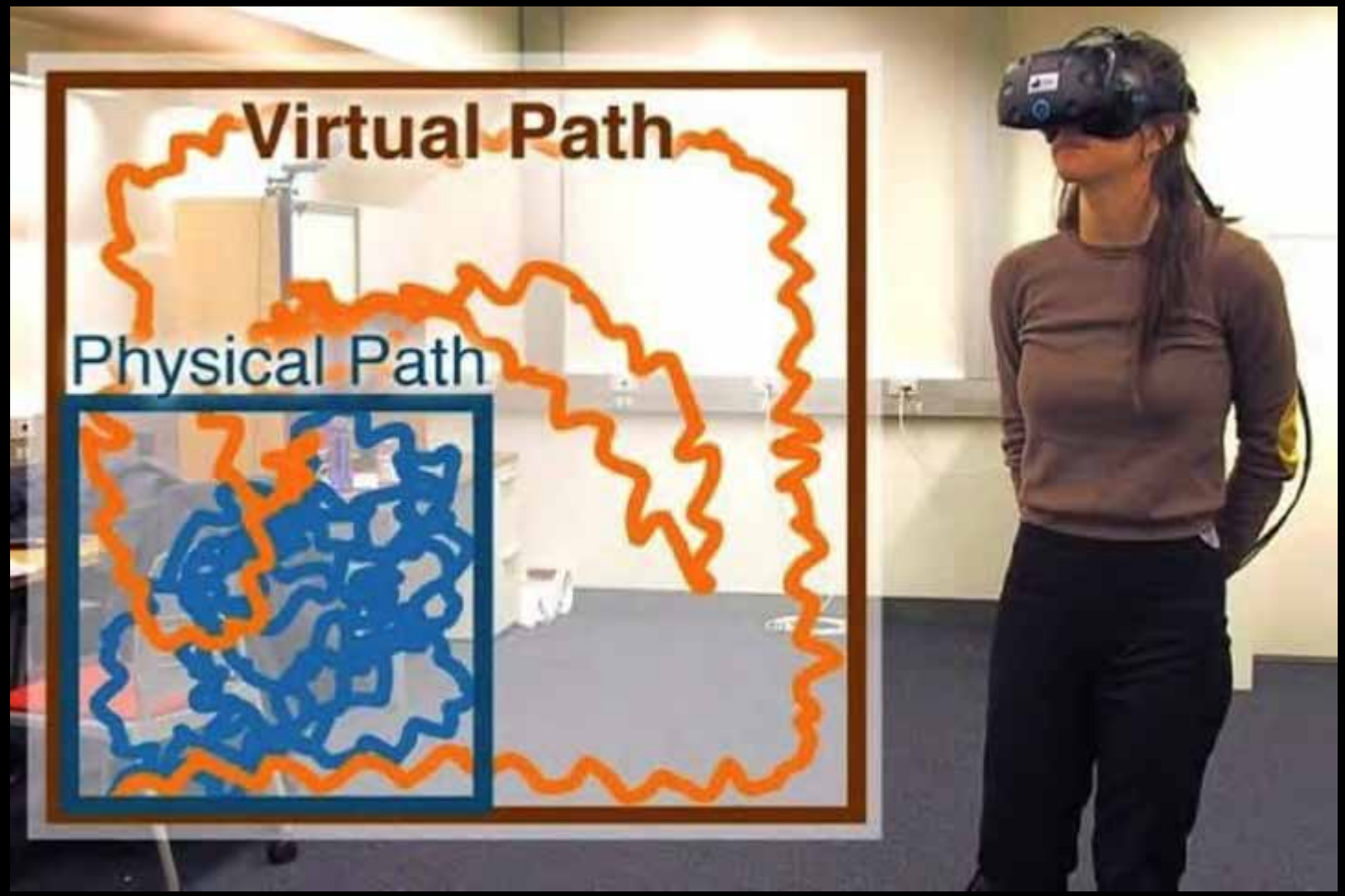
VR Design Specifics: Tracking Saccades will change room play

Saccades refer to the rapid eye movements that occur as you scan a room.

Stony Brook University, Nvidia, and Adobe figure out a way to redirect the VR world as you saccades were occurring.

This allows perceived movement across a large space that occurs in a much smaller space.

This eliminates the simulation sickness often caused by movement.



Questions??