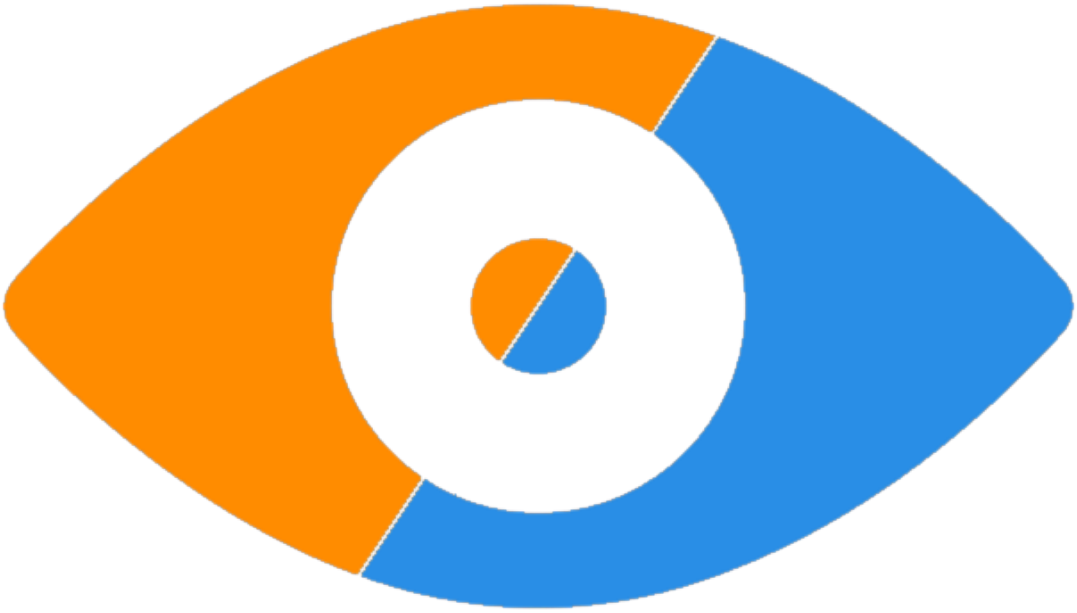


Double Vision

Team Untitled

Shicheng Chu
Eric Lin
Chenhui Pan
Chuan-Wei Sun
Yifei Zhao

Table of Contents

Table of Contents

Introduction

- Gameplay
- Inspiration
- Art Style
- Technical Specifics

Key Game Components

- Basic control
- Special abilities
  - World Switching
  - Lens
  - Portal
- Gimmicks
  - Laser/Mirror
  - Slingshot
  - Moving Platform
  - Key/Door
  - Mud
  - Flamethrower
  - Ice Cube

Technical Features

- Save System
- Cutscene

- Graphics
  - Two Worlds Vision
  - Lens
  - World Switching Effect
  - Portal
  - Outline

Assets

Code References

Highlights

- Baseline/systemic features
- Level features
- Gameplay features
- Graphics features
- Sound features

Credits

Introduction

Gameplay

Double Vision is a first person puzzle game. In each level there are two worlds which overlap with each other. Also they have different walls and other level design. For example, a wall may exist in one of the world only. The two worlds have different art theme, such as larva and ice. The goal is simple. The player must reach the goal point in each level with the help of the special abilities to travel between worlds or generate a portal to swap the substance of the two worlds. With these abilities the player can explore and find difference between the two worlds, and combine the special abilities with gimmicks to finish the level.

Inspiration

This game is inspired by the game Dishonored 2, which is a first person stealth game. In one of its mission, the main character has the ability to do time traveling and use a device to show the vision of the other world. This design is extremely interesting and fascinating, and this mission is also highly praised by the game critics and players around the world. The rudimentary goal of the project is to implement this visual effect and create two worlds that the player can switch between them. We extend the two world idea further and make more interesting mechanics and puzzles and finally let it becomes a first person puzzle game.

Art Style

Double Vision adopted low-poly art style. The decision is mostly because the team members don't have any art-related experiences. The only source of assets are mostly from Unity assets store. This limitation forced us to use low poly style such that the art style be roughly the same between different assets packs.

Technical Specifics

Double Vision is developed by using Unity version 2017.1. The scripts of the game is in C# entirely because Unity does not support Javascript anymore. The rendering path for the entire game is Deferred Rendering to enhance visual fieldiety. Moreover, Ambient Occlusion, Screen Space Reflection, Motion Blur, Bloom, are Depth of Field are used to exploit the data generated from Deferred Rendering. Finally, the whole game is developed in 4 months.

Key Game Components

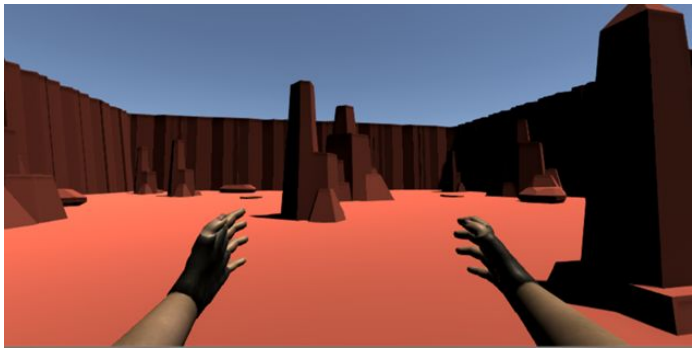
Basic control

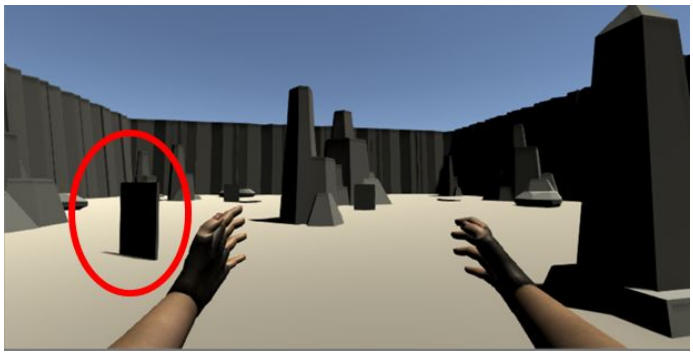
Wal/Sprint:
Player can walk or sprint in the level as most FPS game do.
Jump:
Player can walk or sprint in the level as most FPS game do.
Interaction:
Double Vision only use one button to let the player interact objects in a level. Most of the gimmicks listed below can let

Special abilities

The core gameplay feature of Double Vision is composed of three parts: Lens, World Switching, and Portal.

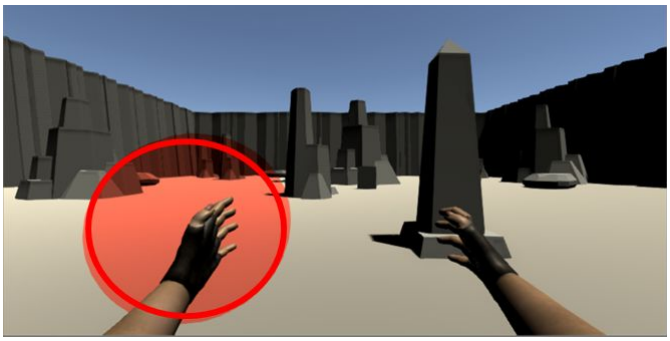
World Switching





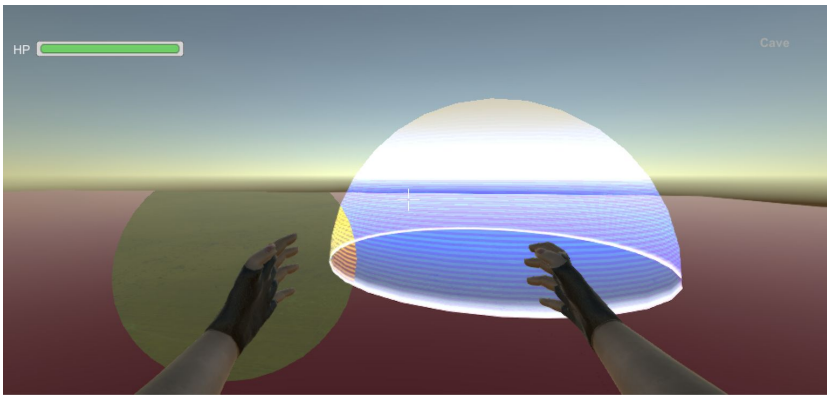
To let player interact with the two worlds, world switching feature is provided. After hitting world switching button, player will see a special visual effect that a boundary is growing outward from location of the player. The objects which wasn't in current world will start showing up as the boundary grows. This visual effect is aim to provide the illusion of the merging of two worlds instead of just switching the vision to the other world. After the switching effect being done, the player will be in the other world. Any colliders such as walls, doors in the other world will interactable with the player. On the other hand, the objects in the previous world will no longer interact with the player.

Lens



Although the player can switch between worlds, he may be constantly switch back and forth around the worlds to find key items or solution to the puzzle. This process is extremely tedious and annoying. Games such as Assassin's Creed has such feature will force player to do the switching constantly. To prevent such issue, Lens are provided in Double Vision. In the game the player has a lens which served as purpose of displaying the vision of the other world. This lens is in shape of crystal and it has its own visual effect such that the player will know that what objects are in the other world.

Portal

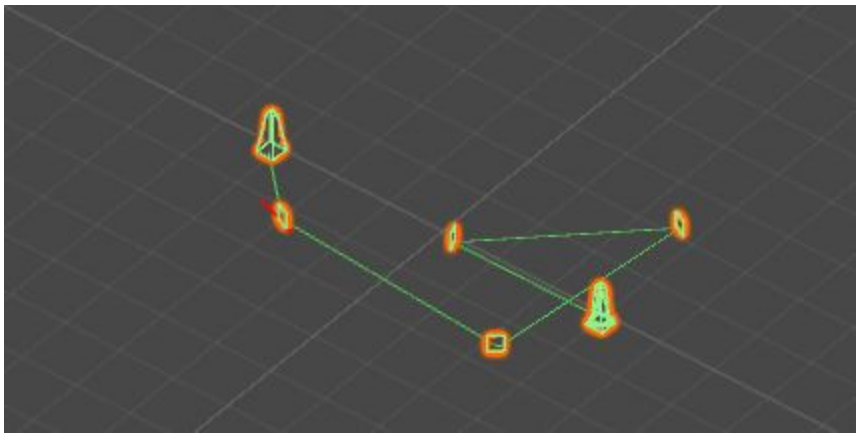


The player can switch between worlds. However, in the perspective of gameplay and level design, switching between two worlds cannot allow us to make enough decent puzzles. All the player can do is switch to the other world and walk pass through a wall. To add more delicate puzzles and levels, portal is added to the core features. The purpose of portal is to let objects in different worlds interact with each other. Also, this feature also allows player to interact an object that he/she cannot reach. In the perspective of game design, this feature is also an attempt to extend the “two worlds” concept: the player now can merge two worlds partially. When the player shoots a portal bullet, once the bullet hits a surface, it will generate a sphere that will swap any substance within the portal to the other world, and vise versa.

Gimmicks

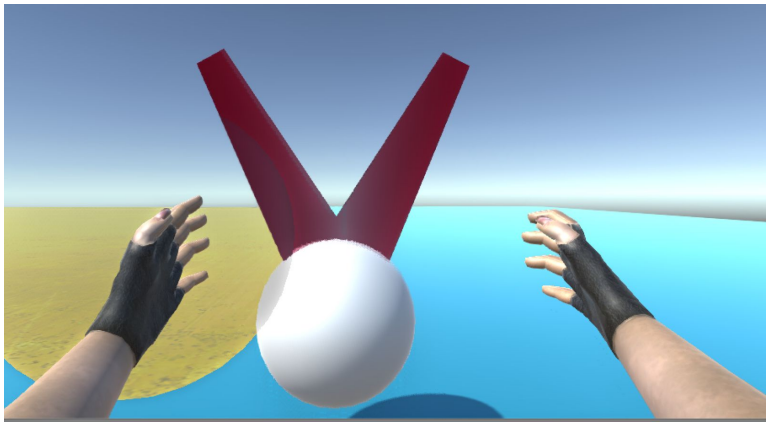
Most of the gimmicks listed below are interactable with player. Many puzzles in Double Vision is made from them.

Laser/Mirror



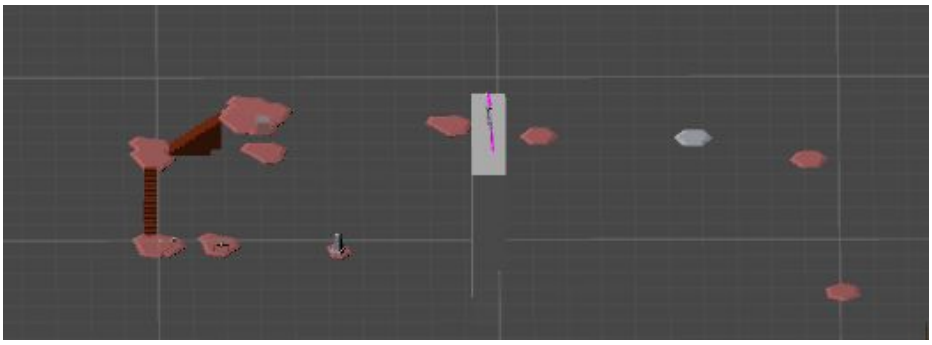
Laser puzzle is widely used in many games such as The Legend of Zelda. When the player turns on the laser, a green beam will show up. The goal of a laser puzzle is to let the laser beam hit the trigger. Once the laser beam hit it, the puzzle is solved, and some new events such as door opening will fire to guide the player to the next area. To further extend the puzzle difficulty, when a laser beam hits a mirror, the laser will reflect the beam. The player can rotate a mirror to adjust reflection direction of the laser beam and let it hit trigger or next mirror. Anything other than a laser will not reflect the laser beam.

Slingshot



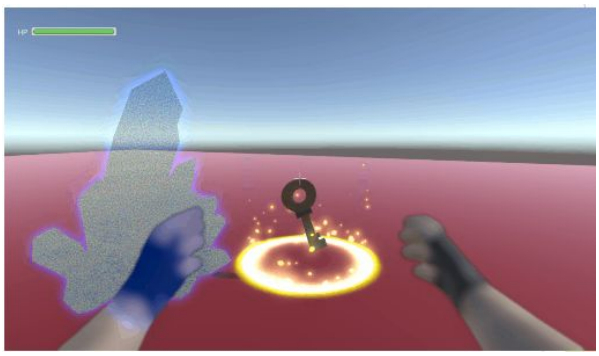
Slingshot is a gimmick that allows player to shoot a bullet through aiming. Once a bullet hits a trigger, the puzzle is solved.

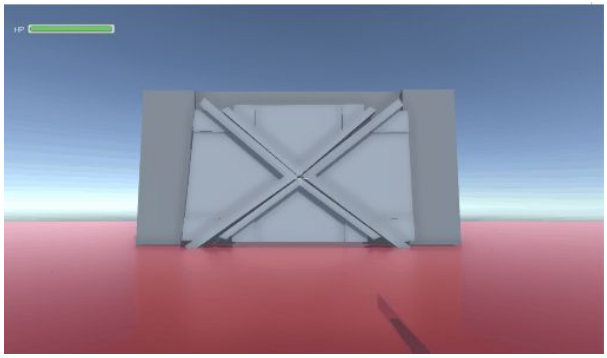
Moving Platform



Moving platform is a classic design in many platformer games. The platform will move and rotate by a given route. Platform is designed to be flexible and easy to use. We can specify the moving speed by using animation curve. Moving direction can also be changed dynamically.

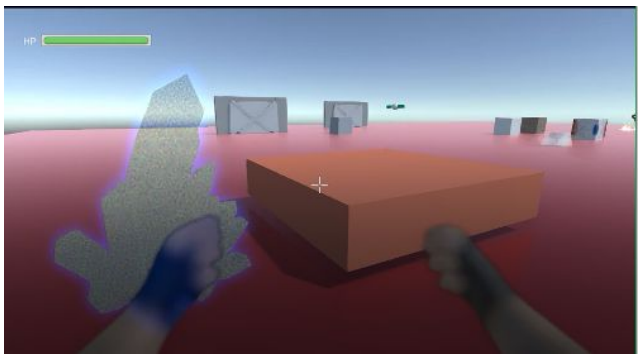
Key/Door





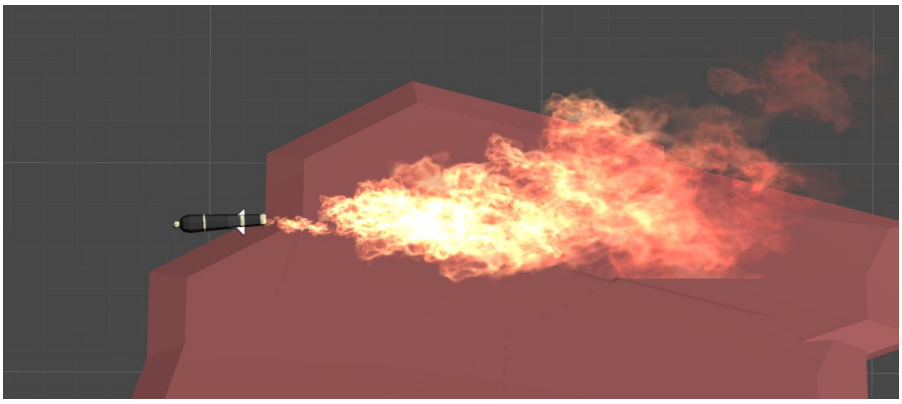
The purpose of key and door is simple: to provide the player a goal to finish the gameplay area. By placing the key at a visible but unreachable place, the player are expected to get the key and open a door to get into next area.

Mud



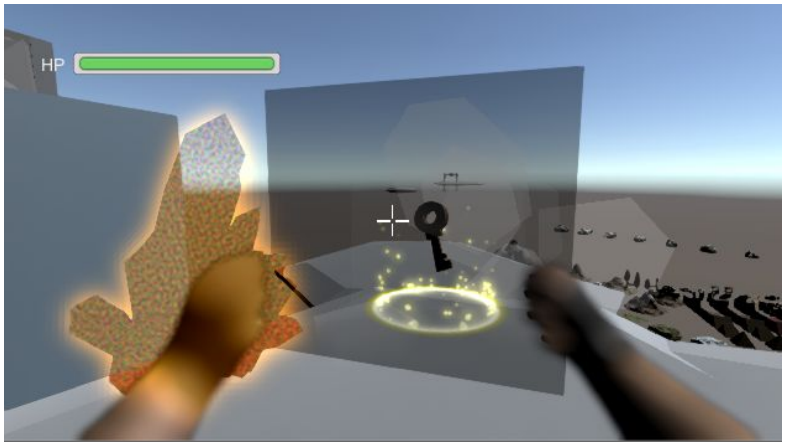
Mud will slow down the moving speed of the player. It will be used for timed puzzle mainly. So the player must avoid the mud and finish the puzzle in time.

Flamethrower



Flamethrower exists in larva world mostly. This gimmick will damage the player if he/she is inside the range of a fire. It will force the player to switch to correct world to avoid damage.

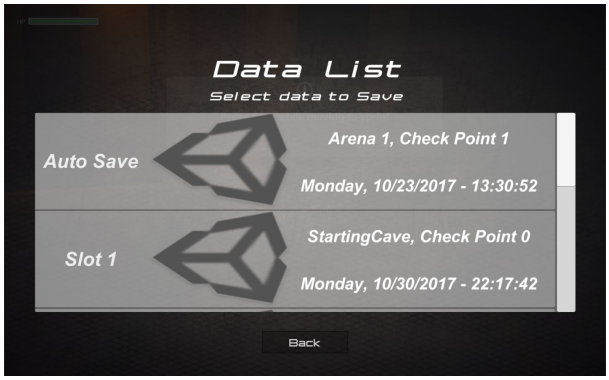
Ice Cube

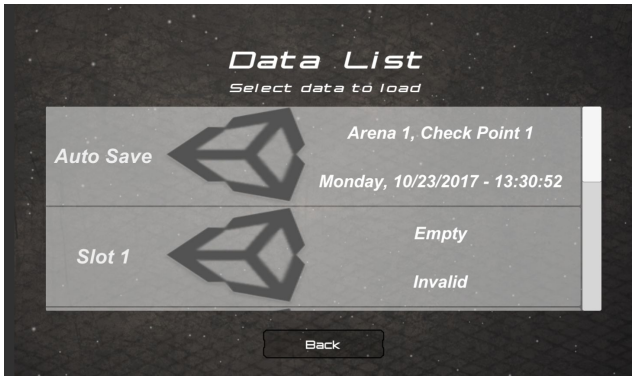


Contrary to flamethrower in larva world. Ice cubes exist in ice world mostly. It will be served as a block to stop the player from moving forward. Also a key item may also be inside an ice cube such that the player cannot access it directly. To remove an ice cube, the player must use flamethrower to melt the ice.

Technical Features

Save System





Double Vision supports auto save. Each level in Double Vision is divided into chunks. Once the player finish a chunk of level and proceed to the next one, it will trigger a checkpoint which will save the progress of the player automatically. A system notification will shows up to remind the player that the game is saved. If the player is dead or quit the game, he/she can start from the last checkpoint and keep playing. Also, Double VIsion provides manual saving feature. The player can choose which save slot he/she wants to save.

Cutscene

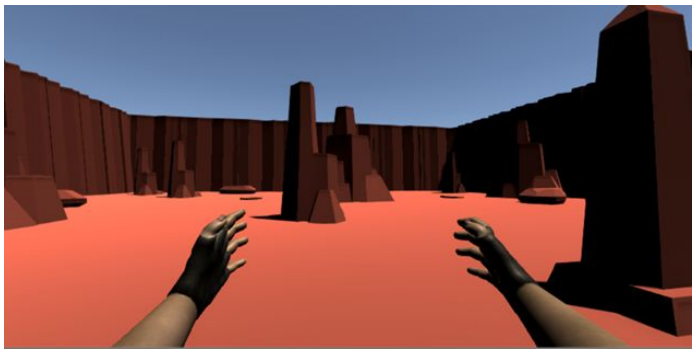
When the player gets a key item and some events happens after that, the player might not be able to know what happened if no further information is provided. To guide the player, cutscene feature is provided. A cutscene will contain important messages that let the player know what to do next. For example, after acquiring a key, a cutscene will be played, and it will show that a door in opening.

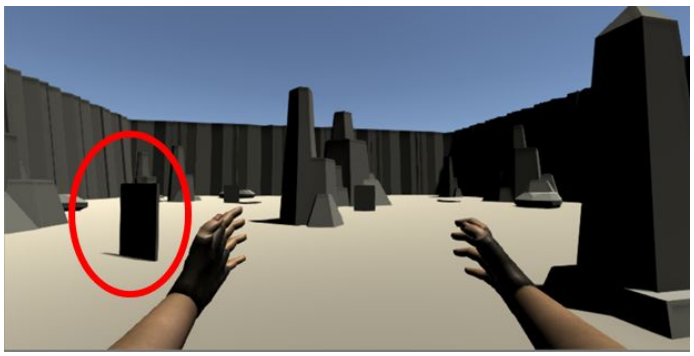
Graphics

This section describes how the core graphics features are implemented.

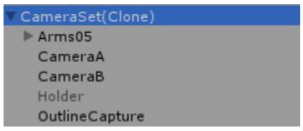
Two Worlds Vision

The goal of the effect is to specify an object which world it is belonged to. When the object and the player is in the same world, it will be shown on the screen. So in this case, take the slab for example, it only be shown in one of the world.

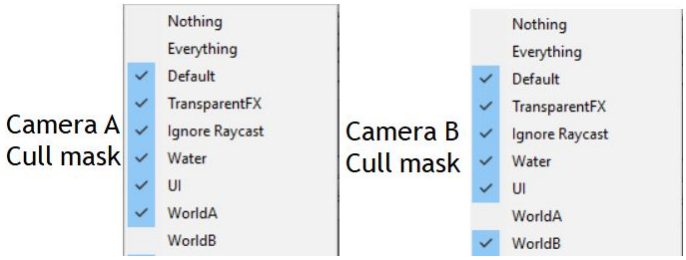




To realize this feature, first we need to specify which the object it's belonged to. The layer of an object can be set to default, world A or world B. Next, for the player, there are two cameras attach to it, which are camera A and camera B. As the name suggests, camera A renders the objects in world A only or default, and vise versa. When the player is in world A, camera A is set to the main camera, camera B is set to the idle camera.

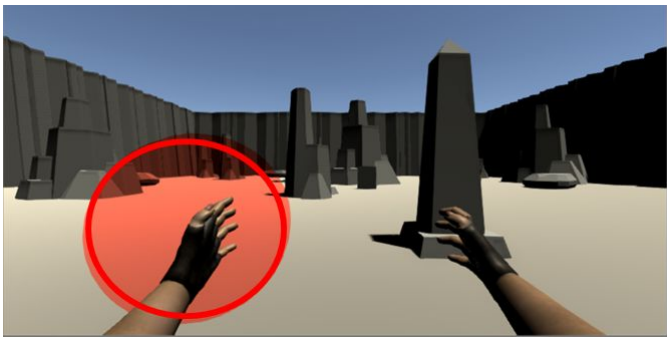


Camera set instance

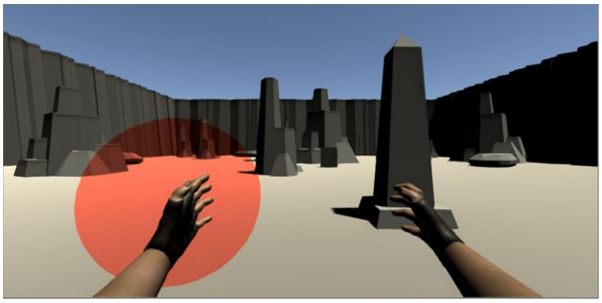


Lens

In a given specific area, we would like to display the vision in the other world through the area. This effect is implemented through general surface shader.



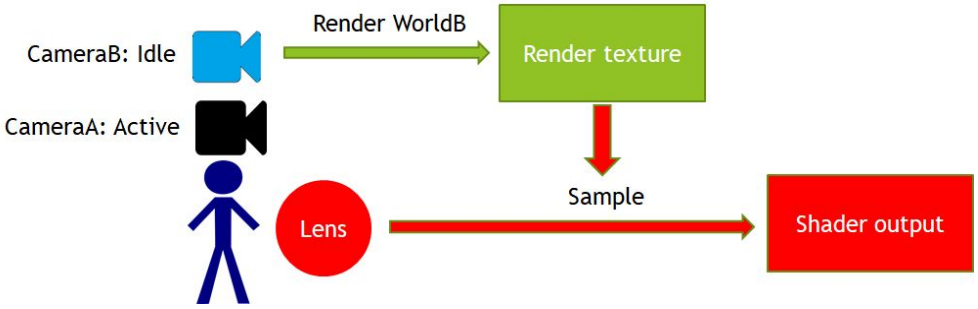
To exploit idle camera (the camera that's not rendering to screen) information, we use render texture technique. When rendering the lens object, we exploit render texture of the other world through the idle camera. Essentially it works like cutting off the matching area in render texture and paste to the lens area.



Main camera: render to screen

Idle camera: render to render texture

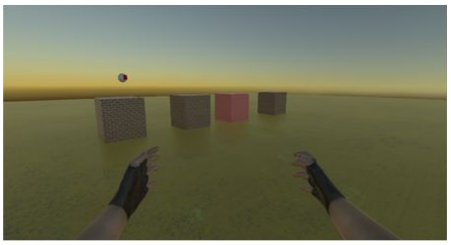
In the example scenario, the player is in world A, so the active camera is camera A. Camera B will render world B, and update the render texture for each frame. When rendering lens, we use a custom shader for the object. This shader samples the render texture through its screen space coordinates, and output the sampled value to final color output.

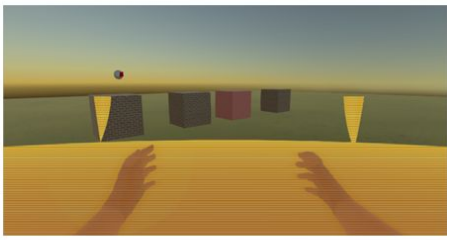


World Switching Effect

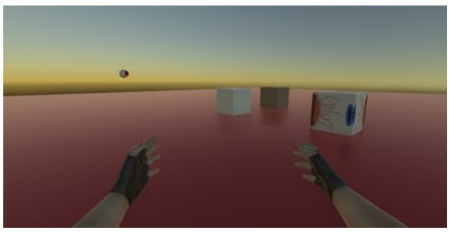
The goal of world switching effect is to have the illusion of merging the view of two worlds. The image below shows the main idea of this effect. To implement this image effect, we'll setup a radius for the shader, and render the effect according to the radius. We also need to animate the transition. All we need to do is updating the radius for each frame. Finally, this effect is done through post processing, because we'll need render output.

Active camera image

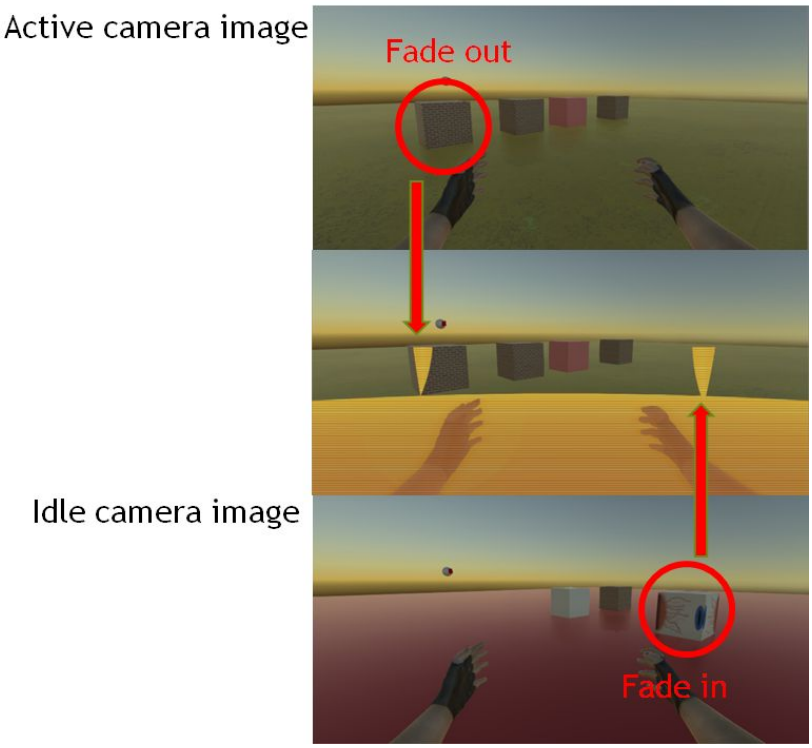




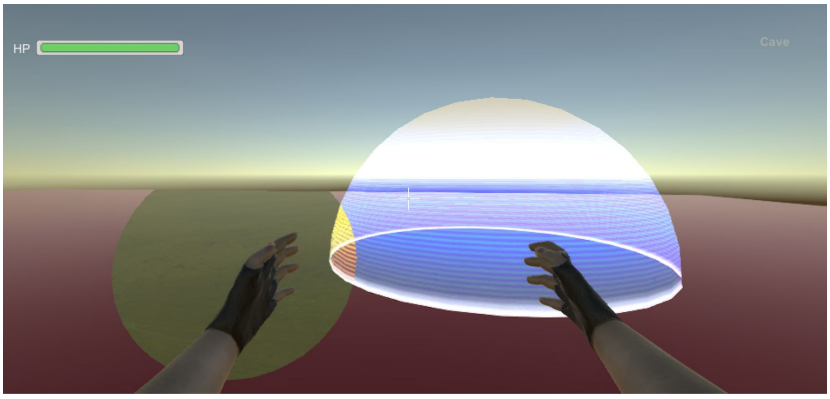
Idle camera image



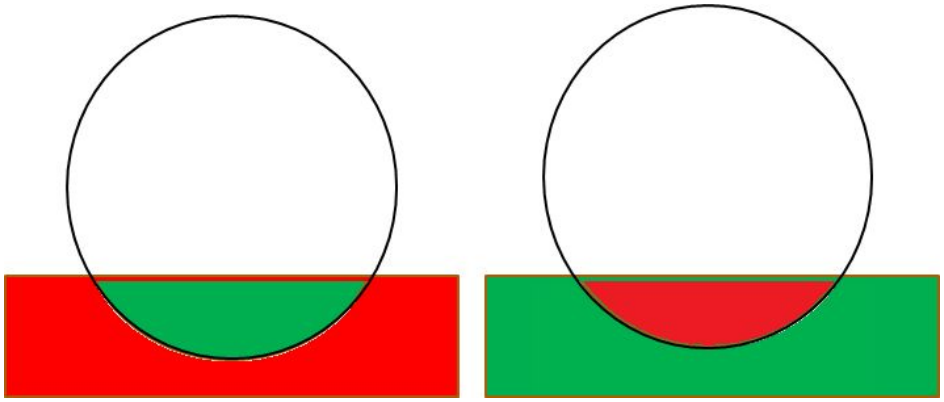
In the implementation. We have radius as input. For each pixel, we reconstruct world space position of main camera and idle camera image from depth textures. Then in the image effect shader we compare the distance, if distance of active camera is inside the radius, we'll do fade out. On the contrary, if distance of the idle camera image is inside the given radius, fade in will be performed.



Portal

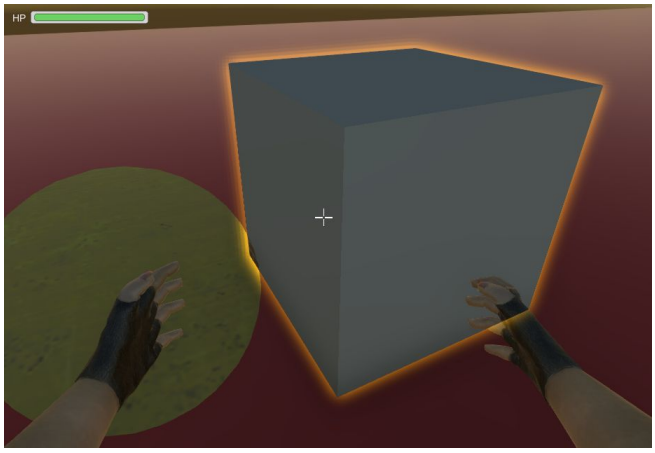


The portal visual effect is to swap the substance between the two worlds. Objects which are inside portal will be transported to the other world. The following image shows the portal effect.

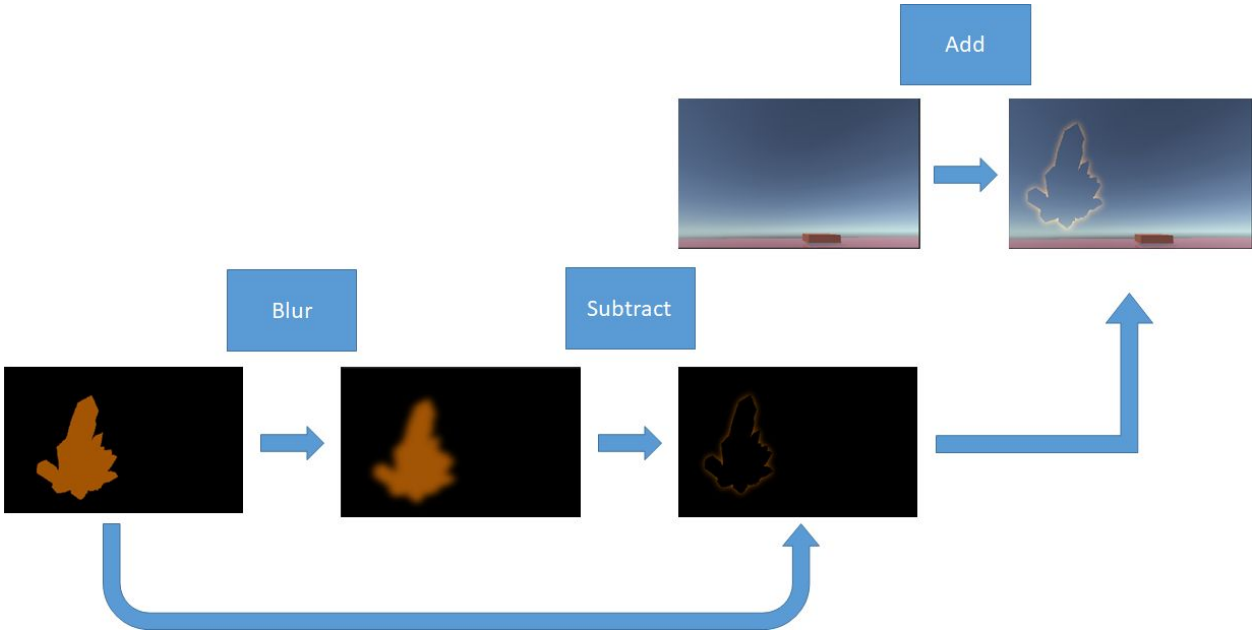


To realise the feature, a shader function called clip is used. This function take a floating point value as input. If the input is small than zero, current pixel will be discarded. For Portal effect, the world space coordinates are used again. The shader will take two uniform variables, portal radius and portal center in world space. If the object should be only rendered inside the portal, the clip function will discard pixels which are outside of the portal sphere, and vise versa.

Outline



Outline is a commonly used visual effect to give hint for the player. It may informs the player what is the key item in the scene. Gimmicks such as laser will use this feature to inform the player that it's an interactable object. The implementation of online can be divided into four passes: Sample, Blur, Subtraction, and Combination.



In the sample pass a camera which will sample objects with outline and render them to a new rendertexture. With the render texture blur pass will make use of a gaussian ping-pong shader to generate a blurred texture. Then the blurred texture will subtract the sample pass pixel value from blur pass pixel such that only blurred outline is left. Finally the blurred outline texture will be added to the main texture.

Assets

3D Games Effects Pack Free
3LE Low Poly Cloud Pack
Action SFX Vocal Kit
Cinemachine
Elemental Free
Fantasy SFX for Particle Distort Texture Effect Library
Fire & Spell Effects
Fork Particle Variety Pack FX
FREE Casual Game SFX Pack
Free Laser

Free Sound FX
Handpainted keys
Lens Flares
Lava Flowing Shader
Low Poly Free Pack
Low Poly: Free Pack
Low Poly Lava World
Low Poly Rocks Pack
Maze Generator
Medieval Town (base)
Nature Pack (Extended)
Particle Collection SKJ 2016_Free samples
Polyon-Adventure Pack
PopcornFX Discovery Pack
Post Processing Stack
Save Game Free - Gold Update
Simple Modern Crosshairs: Pack 1
Sky FX Pack
Stylized Crystal
Unity Standard Assets
Video Capture
Water Effect Fits For Lowpoly Style
Water FX Pack
Water Fx Particles
Eye by Edward Boatman from the Noun Project
Fantasy Music Lite
Survival Shooter Tutorial
Fantasy Ambient Music Pack

Code References

The code referenced from some open source repositories are listed below. However, they're only used partially, and with some modifications.

Transparent object intersection highlight

Gaussian blur shader

Screen space stripes

Highlights

Baseline/systemic features

- Main menu
- Option menu
- Manual menu
- Gameover menu
- Pause menu
- Save/load feature and menu
- Auto save
- Loading screen
- Credits
- Logo
- Trailer video
- In-game UI messages
- Simple cheat codes

Level features

- Two worlds for each level
- 5 levels

Gameplay features

- FPS camera and arms
- Smooth camera control
- Health and Overwatch style (shaking when falling to ground) UI
- Camera shake for falling and damage reaction
- Crosshair
- World switching
- Portal
- Transportable objects
- AI character / agent
- Cutscenes
- Checkpoints
- Slingshot
- Pressure plate
- Laser and mirror
- Lever
- Lock and key
- Fireball
- Flamethrower
- Ice cube
- Mud
- Object carrying
- Trigger buttons
- Moving platforms

Graphics features

- Particle effects
- Portal sphere effects
- Portal clipping effects
- Lens and floating effects
- World transition effects
- Outline effects

- Post processing effects
- Lens flare effects

Sound features

- Sounds (footsteps, guns, portal, world switching, slingshot, etc.)
- Ambient sound for the two worlds

Credits

The Ohio State University, CSE 5912 Capstone Design: Game Design and Development, Fall 2017.

Lecturer: Dr. Roger Crawfis

Member of them Untitled: Shicheng Chu, Eric Lin, Chenhui Pan, Chuan-Wei Sun, and Yifei Zhao