An Exploration of Game Design: Immersive Third-Person Portals

Steve Hughes
University of Technology Sydney
Building 11, 81 Broadway
Sydney, Australia
steven.j.hughes@student.uts.edu.au

Abstract
This paper presents a new approach to the application and use of portals in game design, from the perspective of third-person. To accomplish this, we studied the existing implementations of portal technologies in games, their approach to portal level design and the impact on player experiences regarding immersion in a first-person setting.

During the development of the portals, we encountered tougher challenges than initially expected, leading to a study equally focused on the design and development challenges alongside the study of immersive experiences of players through third person portal interactions.

This project was set in motion not necessarily to provide a better alternative to existing portal games, but to further analyse and assess the possibilities of portals in games. This was achieved by diving deep into the design challenges faced when implementing portals and the divergence from that of first-person to third-person.

Keywords
Immersion, Interaction, Portals, Game Design, Third-person, Camera techniques, Rendering, Perspective

I. INTRODUCTION

Portals are a rare but fascinating video game mechanic that is not only complex to develop, but also complex to implement successfully. Portals connect two areas in a world through a single gateway in a similar fashion to the wormhole theory, though portals generally show a clear view of the other side, just like a looking through standard doorway.

Portals operate by dynamically rendering a texture showing the perspective of the connected portal from the players point of view. This texture needs to be updated constantly as the player moves around the world, always showing the other side from the correct angle and rotation. Placed in a real world setting, this is the difference between looking through an open door or window when compared to observing a framed photograph on a wall. Multiple complexities exist just within the rendering stage, before beginning to evaluate the intricacies of handling the teleportation of players and objects between two connected portals. This paper will outline a number of approaches and design choices used in our prototype in order to overcome many of these challenges, whilst taking a more technical look into the complex camera and rendering techniques utilised by both first and third-person portal games.

This project set out to uncover many of the design and technical challenges involved in developing a game which utilises portals whilst also assessing the feasibility of portals in the third-person, observing the variance of complexity between both first and third-person. Our prototype explores the creative use of third-person portals, showing variance in use cases from traditional first-person portals. We also intend on achieving a similar level of immersion so that player experiences during interactions with portals would not be jarring [5]. This paper further analyses cases where players experience a sense of removal from the game due to these interactions in order to develop a simple set of requirements for successful third-person portal implementations.

Our prototype features a fixed third-person camera implementation, such that the position from the player is generally kept at a constant distance, unless direct line of sight to the player avatar is broken. Fixed cameras are also one of the simplest implementations with the least possibility for alternative immersion-breaking experiences to occur [6]. We proceeded with this implementation on the assumption that this would allow much more development time to be allocated towards analysing the complexities of portal development rather than resolving immersion issues created by external factors.

In section 2 we will cover some related works in the portal game genre, examining their approach to design and implementation. Section 3 will outline methods and techniques used in our prototype, whilst also diving deeper into the technical aspects of rendering a realistic looking texture and handling the complexities which arose during the development. Section 4 will discuss our findings, in particular what we believe to be the most challenging design restrictions that are innately imposed upon third-person portals. Section 5 will briefly outline a summary of our findings whilst Section 6 will explain some of the limitations we encountered during this study.

II. RELATED WORK

In this section, we will give an overview of existing games in the portal genre and examine how they approach their design and implementation of portals, outlining what we believe lead to successful results, giving reference to a simple set of observed rules.

A. Portal Games

Portals were first introduced at E3 in 1998, where *Prey* revealed the concept to the world in their game reveal trailer [1]. Portals gained a lot of popularity much later on in the video game industry when developer and publisher Valve released one of their most successful titles, *Portal*, in October of 2007. Portal revolutionized how portals were used in games by giving the player control of the placement [4]. We have taken many design implementations from Portal in our prototype, attempting to recreate similarly structured portals for players to experience in an alternative perspective.

Prey was finally released in 2006, but utilized portals in a far less exciting manner. Portals in Prey were static objects in the world that existed purely for level traversing and just to show off a new, flashy technique [3]. We make the assumption that these portals had little hinge on the design of their levels due to being such a new technology with a combination of vast design complexity and little experimentation in the field to push boundaries.

The visual quality of games back in the mid-2000s were also of a much lower fidelity than compared to current day games, so we also make the assumption that immersiveness of games today at a visual level are far more noticeable than titles such as Prey. However, Portal made big leaps forward in graphical quality at the time; as such we consider this a valid baseline to compare our levels of immersion.

Finally, Antichamber offers an alternative take where immersion is secondary to puzzle solving. Antichamber takes an approach where the game sets out to confuse the player with impossible geometry and physical scenarios, challenging players to overcome confusion in order to solve puzzles [2].

B. Prototype

One of the main goals of our prototype is to achieve the retention of player immersion, as we make the assumption that disorientation in third-person will create an unpleasant experience for players, possibly leading to frustration at input, camera or movement capabilities as well as the potential for motion sickness [7].

As we also seek to reveal many development challenges in portal creation, many tutorials have appeared since the release of Portal online, especially with the public availability of both Unity and Unreal Engine 4. Many of these tutorials were analysed during the development of our prototype to research different methods of portal creation, and techniques that can be applied to overcome or mitigate the impact on the player experience.

One such tutorial is Sebastian Lague's Unity tutorial, "Coding Adventure: Portals" on his YouTube channel [8]. Despite creating our prototype in Unreal Engine 4, Lague's dedication to researching different structural approaches to portals was a big influence on how rapidly our prototype was able to come together. Lague also created many elegant visual aids when explaining the design side of his development, of which we reference multiple in Section 3.

Whilst Lague's design approaches are fantastically articulated, they are heavily influenced by the same first-person perspective as many of the other existing portal implementations. This led to many additional challenges and restrictions upon both level design when utilizing portals and the technical requirements when developing the components.

III. METHOD

This section outlines the methods used in development and implementation of portals in our prototype, challenges faced during technical development which directly affect player immersion and subsequent design rules which deviate from first-person portal implementations.

A. Resources

Our prototype was built using Unreal Engine 4, using some assets provided in the VR-Basic template. Player avatar and animations were downloaded from Mixamo [10], but no additional libraries were added to assist in the code or design construction. Additionally, the portal was constructed using a combination of both Blueprints and C++, where the complex render pipeline restructuring was handled in C++.

B. Portal Actor and Rendering

The portal actor, or object that exists within the world to display said portal, is required to be large enough for players to fit through whilst thematically making sense. For this reason, we chose a simple door frame with neon highlights to suit the relatively simple geometry that existed within the VR-Basic template. We believe using

a door frame also subconsciously indicates to the player that by travelling through a door, they should expect to end up in a different scene or room to which they are currently situated.

After creating the model for the portal door, we placed a double-sided plane which filled the remaining space within the frame. This plane exists to project an image of the other side to the player so they can see where they are travelling to, to which we will refer to as the *portal plane*. The next step we took was to attach a camera which would rotate around the portal, mimicking the exact movements of the player's own camera. These camera movements would only occur on a portal which is connected to a portal which the player is looking at. This camera would subsequently render its view to be displayed on said portal which the player is looking at. The camera rendering the view should ignore the portal plane from its own side so we can see everything that is hidden behind it.

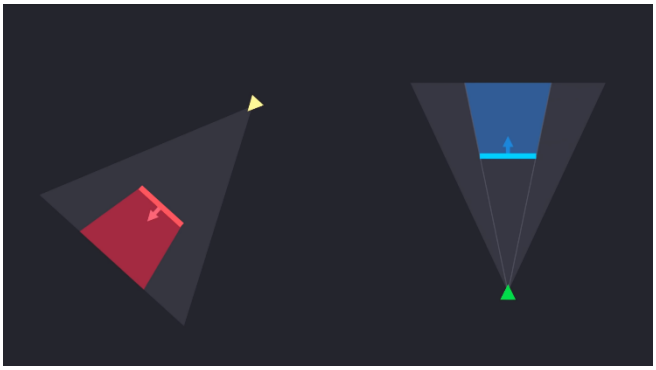

Figure 1: The player is shown in green, the camera of the connected portal shown in yellow. The player is looking through the blue portal, but instead of seeing the blue shaded area, they see the red shaded area. Retrieved from https://www.youtube.com/watch?v=cWpFZbjtSQg

We devised an analogy that allows this method to be visually understood at a very basic and real-world level. Whilst keeping figure 1 in mind, take a picture of two different areas of the game scene with the portals in view from the same angle, one of these being the player's view, the other is the opposing camera's view. We stack these two pictures on top of each other, meaning that the two portals are in exactly the same place. If we were to cut a hole in the players view - the topmost picture - where the portal door frame is, we would be looking out the door frame of the other portal. This works by finding the 4 points of the portal plane which the user is looking at, converting them to screen space, and then mapping the screen space coordinates to the texture which has been generated from the other camera. Only this part of the texture is drawn on to the portal plane in front of the player.

C. Rendering Complications

As we continued testing, we realized a few additional steps were required. In the case that an object exists between the camera and the portal, either part or all of this object will also be included in the render which gets displayed on the portal view in front of the player. In relation to figure 1, this is in the case that an object is present in between the camera (yellow triangle) and the red portal. The resulting textures drawn on our portal views were very disorientating and completely removed all immersion.

To combat this, we decided to experiment with Unreal Engine's clipping settings, specifically the near clip plane. This allowed us to inform the camera to only begin rendering objects that were at least a defined distance away from the camera. The near clip plane is traditionally used to ignore the rendering of blocking elements between a camera and a player avatar in third person, so that players can always see what they're doing. Our solution was to find the distance of the bottom-centre of the other portal to the camera, drawing a plane along the axis which the portal stands and rendering only what was in front.

Figure 2: The other portal camera is rendering one frame behind. The texture displayed on the portal we're viewing shows visual artefacts when the player moves their camera quickly, in this case showing the front of the other portal.

The next issue we began to notice after rigorous testing was the resulting textures always appeared to be one frame behind, as shown by figure 2. Gradually this became more and more noticeable, and given our goals were to retain player immersion to the best of our ability, this was something we believed needed to be addressed. Every time the player turned quickly, there was a noticeable delay in updating the doorway of the connected portal resulting in one thinner and one thicker side of the doorway. The less noticeable artifact was skipping one frame forward as the player walked through the portal. However, same frame rendering turned out to be much more complicated and required further research into the render pipeline of Unreal Engine 4.

We then proceeded to create a C++ class for the portal in order to have access to these deeper rendering methods. This class needed to be able to modify the attached cameras so that we could manually control the timing of their movements and the timing of their rendering. We instantiated new scene capture components as the cameras and disabled the 'Capture Every Frame' option. The other requirement was to create a new camera class which was attached to the player, so we could override the rendering function of the player's camera, which resultantly draws the players view to the screen.

Figure 3: The red players camera position is shown relatively on the connected portal with the red arrow. The green player is on the other side of the connected portal in the far background, but appears as though they are on the other side of the portal frame.

Every frame before the player camera takes a snapshot of the scene, we iterate through any visible portal, tell the connected portal to move their camera to the same location as shown in figure 3, capture their scene, render it to the texture, then have our visible portal update the texture displayed on its portal plane and only after all these steps were complete would we allow the player camera to take it's snapshot.

A few final touches to complete the polish of our portal from a rendering stance were required. Enabling post processing on the connected portal camera was necessary to fix colour correction, bloom and anti-aliasing artifacts. Converting the plane to a very thin box and enabling two-sided rendering of the texture was another small tweak to remove any frame tearing experienced when travelling through the portal.

D. Teleport Handling

To complete the two main functionalities of our portals, we needed to develop a solution which moved the player from the current portal to the connected portal at the right time, to the right place. In order to achieve this, we created two trigger volumes on either side of the portal so we knew which side the player was on. After this, we created an empty component on the connected portal to track the player location, updating it constantly as the player moved.

Despite not being the most efficient solution at runtime, we believe this was a very efficient way to handle rotating and transporting the player transform without getting too caught up in more complex matrix comparisons and modifications. We simply copied the transform of the empty object and updated the player avatar to match when the avatar would cross the centre of the portal from the entering direction. However, we decided to give the avatar a little buffer on either side, so they were not instantly teleported the moment they passed local X and Z zero coordinates. When the player teleported at this point, they would often get thrown back and forth until they were far enough away. Setting this buffer allowed the player to get far enough away from the centre that it would no longer attempt to calculate a new teleport. Having a slight thickness to the portal plane allowed this transition to be seamless.

E. Teleport Complications

We had our player avatar seemingly teleporting to the correct position and at the correct location, yet the avatar would constantly shift to the side by just a few units every single time. We assumed there was an error in our formula, so to confirm our theory, we displayed the avatar offset from the portal centre both before and after the teleport occurred. However, we were getting identical results, meaning there was no issue with our formula - the player avatar was ending up at the exact same spot.

Upon further investigation and many recorded footage reviews, it became apparent that the faster the player was moving, the further offset they would be. This was the result of incorrectly updating the physics state of the avatar, so their forward motion would carry over to the other side. The avatar was continually trying to move in the same direction they were previously running prior to the teleport, ignoring the rotational change to the player during the teleport.

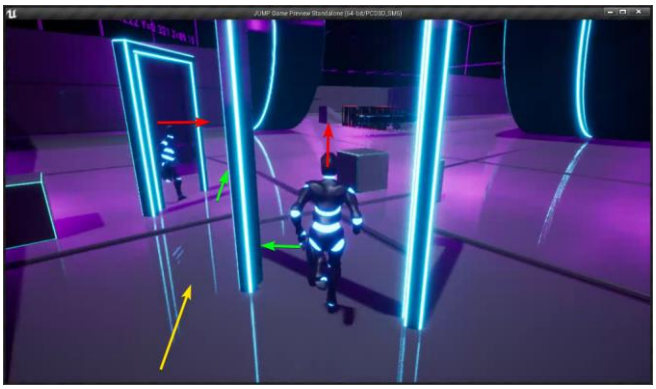

Figure 4: These two portals are connected. The red arrow shows the player's current velocity, green shows their velocity after the teleport. The yellow shows their velocity in world space, whilst the green arrow above it shows the continuation of that same velocity, hence the reason for their relative velocity shift, post teleport.

After saving the velocity of the player before the teleport, clearing, rotating the saved velocity and applying it again to the player, our movement and teleporting was smooth and seamless.

IV. FINDINGS

In this section we break down our findings, covering the limitations related to the design of levels, general camera principles which heavily reflect player immersion, and summarizing a few points which led to the development being more challenging than that of a first-person implementation.

A. Level Design Limitations

We took another approach of attempting to recreate existing configurations we had seen in the game *Portal* to evaluate the design possibilities of our third-person implementation. Here we encountered a new issue impacting third-person portals only. *Portal* allows users to place a new portal on a wall or flat surface so that they may walk through the portal and appear at the resulting location, which again is simply a hole in another wall.

This is when we stumbled upon a very impactful downside of our fixed, third-person camera implementation. To briefly summarize, the camera is incapable of staying in the same position relative to the player if the other side of the portal is backed up against a wall. The camera will automatically jump forward to remove blocking objects from in between the player and the camera, in this case, a wall. This effect completely ruined any hope we had of seamless transitions between portals, as the player on screen would completely change its relative position. Going forward, most of our portals were placed in a location which did not back up against a wall. A few portals were left in place to demonstrate this effect.

Another level design implementation we wished to include was the vertical rotation of portals themselves. This would allow our level designs to incorporate traversing through sloped surfaces so that more complex puzzles using physics rotations could be pursued. Unfortunately, we discovered this would require far too much time to implement than what we had allocated, but we believe that following the guidelines set here, it should be possible. This would require all vertical rotations of the player to be smoothly adjusted back to a standing position, with the camera also following the avatar's rotation. As the player has control of vertical camera rotation, we assume any input to the camera on the vertical axis should take priority and the continuation of the camera following should be interrupted and discarded.

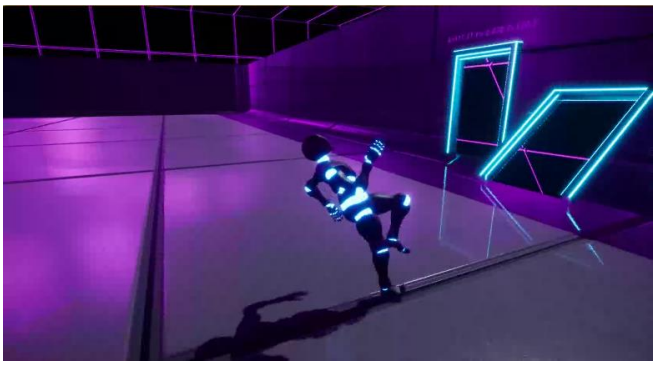

Figure 5: After traversing through the portal shown on the right, the player's body was bent backwards and camera controls were on an off-axis rotation.

B. Camera Limitations

As aforementioned, we believe that a fixed third-person camera is the right choice for any prototypes of portal implementations on a tight time constraint. As we progressed through the development, we also came to the assumption that it is also one of two correct implementations. The other camera setup that we believe will work is a dynamic third-person system [11], where there are special interactions that exist for portals. The camera would need to abandon its existing state and follow a player as close as possible when approaching a portal or giving indication of intentions to traverse through a portal.

This is due to an existing design principle relating to cameras, arguing that players should always be able to see the scene in front of them, or see the scene in their direction of travel in order to avoid disorientation occurring [12]. We believe to have confirmed this to be true for most players, where our prototype clearly shows the other side of the portal where players are intending on travelling. The portal plane, portal frame and player avatar are the only constants which remain when a teleport occurs and despite around 75% or more

of the screen updating dramatically, the portal transition still feels smooth.

Our assumption is that players' peripheral vision is less impactful on their state of immersion, and objects which are the main focus of the players attention hold the most value for retaining said immersion. If these objects are manipulated in an unexpected way, players will feel some sort of disorientation or removal from the game world.

We tried to design our levels to keep our camera system within a few rules that exist in the cinematography world, such as the 30 degree rule [13]. Cinematography rules state that any camera cut which occurs should always rotate the perspective by a minimum of 30 degrees. We noticed that when our camera position was moved in relation to the player, in instances such as colliding with a wall, the rotation stayed constant and the result was very jarring. Whilst we could've implemented a 30 degree rotation in these instances, camera cuts are still conflicting with our idea of a seamless transition, which we believe will create an unnatural sense of motion to both the input and the player's perspective.

V. FUTURE WORK

Going forward, our prototype shows that third-person portals are viable for future game development, but also leaves a lot left to be desired. The complexities encountered during development led us to spend much more time than we had initially anticipated in the development stages rather than diving deeper into solving design issues. In this section we will cover the approaches we believe will allow continued research in third-person portal development to further assess and analyse the extent to which portals can be implemented in games.

A. Camera Systems

As we briefly mentioned in section IV, our assumption is that with a more lenient development timeline, a dynamic third-person camera system with heavy focus on portal teleportation could very well lead to a much cleaner solution than provided in our prototype. If a system were to be implemented where the camera closely follows the player avatar through the portal, either in the form of a cutscene like motion where all player input is locked, or a transition to a much more close-up perspective, traversing portals could bring an incredible experience to the table.

We believe a dynamic third-person implementation is much more likely to suit a storytelling or role playing focussed title due to the emphasis placed on cinematic style camera angles and ability to transition between shots in a similar fashion to film [14]. These camera angles generally remove much of the focus from the player avatar and what's directly in front of them, allowing the player to analyse their surroundings and take in the environment. Any transition where the camera slowly zooms in on an object will lead to a much more focussed state, removing the focus from the environment to the player avatar, the portal and the direction they're moving. These are once again the factors we believe are key in having a smooth portal transition, regardless of whether or not the camera itself actually passes through the portal doorway.

In the case of portals being utilized in creative level design, we believe that a fixed third-person approach is essential to achieving a seamless transition. This could cover a range of genres from puzzle platformers to action shooters. Each of these genres should have their own specific rule sets applied to their camera managers in order to achieve a high level of immersive experience.

B. Level Design

In order to further explore the extents of level design in relation to portal usage, a larger development time frame would once again be required. In these cases, we believe that there is much to explore in our chosen field of platforming, especially regarding their uses in physics manipulations in order to solve challenges. Whilst we were unable to accurately measure whether high-speed traversal through portals against walls negated the jarring effect of camera snapping, we believe that this may be a possible solution.

When assessing the design of levels in alternate genres, we believe there is much more flexibility in the design of both the appearance and physical placements of the portals. In games where fast paced action is likely to be occurring, larger portals in areas where there are fewer obstacles to cause camera collision may be a very simple implementation. We also believe that genres such as exploration, survival, MMOs and even horror games have much to be explored when it comes to level and environment design to suit portal implementations.

The effect of art styles was not a topic we covered earlier in our paper but remains one that we believe can drastically change physical and visual properties of how third-person portals may slot into differing worlds. These near endless possibilities lead us to believe there is much more to learn and much more to discover.

VI. CONCLUSION

Our prototype confirms the feasibility of immersive third-person portals in games yet leaves the limits on just how far this mechanic can be taken, a complete mystery. We believe that with a strong set of rules for camera management and strict design rules for their placement, third-person portals can be implemented successfully in nearly any third-person game where thematically fitting. Immersiveness of the portals is very much tied to the ability of the development team, whether they can successfully recreate the same-frame rendering techniques outlined in the paper or find another solution which minimises the transition jarring and take careful consideration of where they are to be implemented.

Whilst the prototype we constructed was still a great success, we were ultimately left with a feeling of curiosity for what experiences and techniques can be made possible with further development.

REFERENCES

[1] u/onex7805 2019, 'Prey E3 1998 trailer, showcasing the very first portal technology […]', forum post, Reddit, 28 September, viewed 13 June 2020, <https://www.reddit.com/r/ObscureMedia/comments/da4xrr/prey_e3_1998_trailer_showcasing_the_very_first/>.

[2] Gallegos, A. 2013, *Antichamber Review*, viewed 13 June 2020, <https://www.ign.com/articles/2013/02/01/antichamber-review>.

[3] Sooperchikken 2018, '*Prey (2006) doing portals before Portal made them cool'*, YouTube, viewed 14 June 2020, <https://www.youtube.com/watch?v=TZQdS8Uz-jQ>.

[4] Gamehelper, 2006, '*Portal Release Trailer'*, YouTube, viewed 14 June 2020, <https://www.youtube.com/watch?v=TluRVBhmf8w>.

[5] Haigh-Hutchinson. M, 2009, 'Real Time Cameras: A Guide for Game Designers and Developers', *Morgan Kaufmann Publishers Inc*,Viewed 13 June 2020, <https://dl.acm.org/doi/book/10.5555/1611366>.

[6] Schramm. J, 2013, 'Analysis of Third Person Cameras in Current Generation Action Games', *Dissertation*, Viewed 15 June 2020, <http://www.diva-portal.org/smash/record.jsf?pid=diva2%3A628121&dswid=-1335325232>.

[7] Ray Corriea. A. 2013, 'How developers are trying to solve motion sickness in video games', Polygon, viewed 15 June 2020, <https://www.polygon.com/2013/10/26/4862474/video-games-and-motion-sickness-dying-light-techland-fps>.

[8] Lague. S, 2020, '*Coding Adventure: Portals'*, YouTube, viewed 18 April 2020, < https://www.youtube.com/watch?v=cWpFZbjtSQg>.

[9] Adobe, 2020, Mixamo, Accessed 28 April 2020, <https://www.mixamo.com/>.

[10] Pruehs. N, 'Six ingredients for a dynamic third person camera', *Unreal Engine Blog*, April 28, 2018, Viewed 10 May, 2020, <https://www.unrealengine.com/en-US/tech-blog/six-ingredients-for-a-dynamic-third-person-camera>.

[11] Nesky. J, 2014, '*50 Camera Mistakes'*, Video presentation at the Game Developers Conference 2014, <https://www.gdcvault.com/play/1020460/50-Camera>.

[12] International Academy of Film and Television, 2014, '*Directing – The 30 Degree Rule | International Academy of Film and Television'*, Viewed 28 May, 2020, <https://www.youtube.com/watch?v=orsQ5kVvN3Y>.

[13] McIntosh. M, 2012, '*The Cameras of Uncharted 3'*, Video presentation at the Game Developers Conference 2012, <https://www.gdcvault.com/play/1015514/The-Cameras-of-Uncharted>.