Roomfininty - The Biggest VR room in The Smallest amount of physical space

Sunday 15th January, 2023 - 23:19

Leonint Amarantos
University of Luxembourg
Email: leonint.amarantos.001@student.uni.lu

This report has been produced under the supervision of:
Nicolas Guelfi
University of Luxembourg
Email: nicolas.guelfi@uni.lu

Abstract

Virtual Reality(VR) - an amazing Human-Computer-Interaction(HCI) technology that allows physical space to be expanded with computer graphics. This paper seeks to find and analyse different methodologies that can allow to physical physical space limits even further and allow natural walking in VR. Based on existing research 3 currently most viable technologies are explained and evaluated for their use cases, downfalls, and possible future improvement. On the technical side a small 3D engine is developed using JavaScript and WebGL2 API, that conceptually demonstrates how "Portals" - a non-Euclidean concept can be implemented and used to navigate infinitely large VR spaces.

1. Introduction

The following paper seeks to improve the current state of the VR technology. With the current state remaining a niche due to high costs, low acceptability, etc.
As such scope is very large the paper focuses on the particular part of the HCI, which is locomotion or movement of the user in a common, non-specialized environment. Immersiveness is identified as a high priority as it often becomes a blocking point for many VR users. This paper goes over different techniques that make attempts at achieving the desired immersive effect of natural walking in VR. With research being conducted on the existing works.
On the technical side the paper provides an input into the development of the domain through proposal of an implementation of one of these techniques in form of a working 3D engine.

2. Project description

2.1. Domains

2.1.1. Scientific domains. Human-Computer-Interaction(HCI) - A field of study that crosses into multiple disciplines like psychology, VR, UI, etc. and aims to improve design of the technology to be easily interacted with. VR -

2.1.2. Technical domains. Computer Graphics - A field that studies production of an image through computation, this involves anything that ends-up on a display from games to movies.
Software engineering - A field that studies efficient development of robust software through planing, paradigms, best-practices.

2.2. Targeted Deliverables

2.2.1. Scientific deliverables. In the real world we have many limitations which are not present in VR such as space. VR is getting more immersive and realistic, but it is uncommon to be able to interact with it in the most intuitive way for humans - walking, idea which was being developed even back in 1999 [1].
A common user simply will have not enough space for the worlds available in VR. For example, a person inside of his bedroom wants to virtually walk around the Louver museum. The idea can be thought of as mapping of coordinates between 2 Vector spaces, VR and real world. Let the bedroom and the virtual Louver museum be closed spaces of any shape or size on different 2D planes, where any 2 coordinates can be chosen inside of a bounded area and a transform can be applied from one to the other.

2.2.2. Technical deliverables. The main goal of the deliverable is to create and present a proof of concept solution to limited space when interacting with VR. The solution is focused around creating an object referred as a "portal" which can be thought of as a window to a different place, imagine you are at home in Luxembourg and in your window you see the top of the mount Everest, going through it, you will appear on the mount Everest. Using those "portals" it will be possible to overlap the virtual spaces to make fit in a set target of real space.

Thus result of the deliverable is a small 3D engine capable of rendering and allowing user interaction(walking trough) with “Portals” is a top priority and It will have the most development time allocated to it. The plan for the engine to be written in pure JS, rendering using WebGL.
The deliverable also includes a way to present the concept to the user. Thus multiple demos are produced using the tools from the engine. The demos are displayed on a web page in a scrollable list of cards shortly describing the demo, the user is to choose a demo by clicking a “run demo“ button on the card.
The deliverable is produced with JavaScrict language and WebGL API.

3. Pre-requisites

3.1. Scientific pre-requisites

This paper assumes that a reader has an understanding on what is Virtual Reality(VR) in this context and it’s main use in current media. VR is a simulated environment that a user can immerse in using a head-set that displays it on a 3D screen, it is always accompanied with an input interface whether it is a joy-stick, body tracker, accelerometer, etc. Most common use currently is consumption of media like games or movies and in-office training(military, machinery ,etc.).

3.2. Technical pre-requisites

The project will be written mostly in Java Script, however a strong experience in any other C-family language will be satisfactory.

Rendering will be done with WebGL2 api which will be used extensively. Highest complexity will be related to cameras. Therefore it is recommended to have prior knowledge about rendering basics, transformation matrices, and understand vocabulary like view frustum, near-plane, clipping plane. Shaders written will be quiet basic, so no deep knowledge of GLSL is required.

The engine will take form of classical scene graph parent-child relationship, thus any prior use of existing once like Godot, Unity3D, Unreal Engine, etc. is sufficient. Parching of files like .obj and .glTF will be used, thus it is rather good to know what they are.

4. How can large virtual space be made physically navigable in the real world?

4.1. Requirements

The requirement of the study is to explain the problem of physical navigation and interaction with large virtual environments. Then to analyse existing in the field physical VR navigation techniques(will be refereed to as solutions). The paper also should describe a set of key characteristics that can evaluate their performance in comparison to each other.
The production must provide the reader with a clear explanation of the terminology used and present the environments that apply to the question. As the study analyses existing solutions it should be well cited with multiple sources that support made claims.

4.2. Design

The proposed question already has an amount of very different solutions that are at a different state of solving the question, some rely on hardware, some on software, combination of both. Therefore instead of generalising and explaining the problem mathematically, this paper rather takes a form of a review on the current state of technology with a deeper insight into a selection of individual techniques. It is mostly be whitten in a general language with use of domain-specific terminology and references to the source materials. After an extensive search through Google Scholar the following sections are produced:

4.2.1. Introduction to the question. Before any analysis or study this paper provides a list of key-words and concepts important for the understanding of the further content. Then the paper introduces the reader to the question by decomposing it into small sections:

- What is ”physically navigable”?
- What is ”the real world”?
- What is ”a large virtual space”?

This not only prepares the reader, but also gives an opportunity to describe the environment of the problem and the use cases of the study and set restrictions onto the goals to be achived.

4.2.2. Key characteristics of a VR navigation technique. This section introduces several key characteristics that describes a way to evaluate success and viability of any potential solution. This is important as the solutions vary drastically between each other and it might be difficult to evaluate them without a clear goal. It is also important to explain the reasoning for the chosen variables and what will be a gold standard for each of them.

4.2.3. Current solutions. In this section a selection of current solutions that are both on the market and are being developed will be introduced and explained to the reader. They are then going to be evaluated using the variables we have defined earlier with support of visual representation.

4.3. Production

4.3.1. Key-words.

- Virtual Reality(VR) - A space that exists virtually, generated with a computer that can usually be interacted with through a 3D display and input devices.

- Locomotion - Act of moving from one position to another. As the paper seeks to answer the question of physical navigation it implies that the act of movement will arrive from a translational displacement of the user compared to the walking surface, thus it can arise from translation of either the user or the surface.
- Immersive - A deep involvement in a particular action or situation. With VR, immersiveness more often describes a proximity to the reality and simplicity of interaction. An immersive VR world would not necessary be perceived as reality, but would be interacted with as seamless as one.

4.3.2. What is meant for space to be physically navigable and why is it important?. The most popular VR input devices besides the VR helmet itself that captures head rotation of the user are currently:

- Head Mounted Display(HDM) - captures rotation of user's head.
- Remote controllers - tracks position and rotation of user's hands that is later processed to allow for gestures.
- Cameras - preform full-body tracking with or without a suit and additional markers. Which allows for even more complex actions like dancing.

All of them achieve a very intuitive and immersive HCI, where hands are used for gabbing, moving, pointing, and head performs rotation.
However walking is predominately replaced with presses of buttons and rarely comes from the movement of the user, this largely reduces applicable scenarios like training, potential users like elderly, etc. Thus it is a very natural step in development of the domain is to create environment where a most natural for humans locomotion is possible - walking.
What is "the real world"?
This study aims to be applicable for a wide audience, this a targeted user is assumed to be a common person in a household, an office worker, or trainee at a military facility. Said otherwise the paper is focused on environments with limited free space, or that is not specifically built with VR as a sole purpose. Approximation for the total available area range that will be used later in this paper is 10 - 50 m^2.
What is "a large virtual space"?
A large virtual space targeted is any space that can be rendered on modern machines, whether it is a VR game, a VR museum or prototype of a construction work, it needs to be acknowledged that in most of these modern applications the available real world space can be significantly smaller than a one created virtually. Thus "a large virtual space" is assumed to be larger, ideally infinitely larger than the real world space. Therefore the paper is seeking for solutions that can increase the navigable area of the virtual world by magnitude of infinity at leas in one direction or axis of movement.

4.3.3. Key characteristics of a physical VR navigation technique.

- Space required - The minimum area of free space that is required to use the solution.
  It was said that the space of the target user is highly limited and thus it is important for a solution to fit in the earlier discussed range 10 - 50 m^2
- Space gain - The ratio of the total real world space used to the virtual space that can be traversed.
  The proposed question is looking for a solution to at least double the real world area. However the goal is to have an infinite gain, thus an environment of any size could be navigated.
- Safety - What is the potential harm or discomfort of the solution and what precautions are required in plane.
  As the user will be wearing a headset his real world vision and hearing will be nullified, so the immersed individual will be more prone to an accident.
- Limitations\Degrees of freedom - What are the axis the user can move on and what are other limitations.
- Opportunity cost - what is the trade-off off using the solution, whether it is a purchase of equipment or limitation of applicable scenarios.

4.3.4. Physical navigation in VR techniques. As discussed earlier the most used virtual navigation technique currently is a point and click teleportation system, aka. the user points to the desired position and uses a button to communicate movement to the spot. Numerous other stationary techniques that rely on gestures, remotes and simulations [2]. However the once that satisfy proposed in the question restrictions "physically navigable" are: Active repositioning(ex. treadmills), Redirected walking(RDW) and Non-euclidean spaces. It is also important to note that the question stated is about the HCI and performance of techniques is looked at the point where it is viable or does not provide a significant discomfort, rather than undetectable by the user [3].

Active repositioning - On the surface level active repositioning is a human-motion driven treadmill that is used as an input device. However it can take a more elaborate form like a hamster ball, multi-directional treadmill or entire moving floor. Generally active repositioning means that a surface, most likely and practically a looped surface is being either pushed by user or moved in response to the user impulse to move, to create a translation difference and record it as a movement input. The idea to use treadmills in VR is not a new one, [1] however the advancement in this area is mechanically bound and therefore did not enjoy as significant traction as once mentioned later like RDW.
Space required for the solution is going to be equivalent to the hardware size, which is mostly not different to a usual treadmill for sport, which is in the range of 1.5 - 3.5m^2(Space required 1). The space gain from treadmills is

inherently infinite as the walking surface is looped, providing all freedoms of movement in multi-directional case(1), however it is often restricted to one axis movement on classical treadmills(1). When emerging the user into VR, real world capabilities of the user decrease tremendously balance, vision, hearing, etc. therefore there is a higher chance of a potential injury, this critique will apply to the most of the solutions presented, however it is amplified in this scenario as it involves fast semi-independently or independently moving parts. This can be lowered by adding additional clearance between the treadmill and other objects/abstractions like walls, additionally the user can be supported by a harness to prevent falling and tumbling over. Currently multi-directional treadmills do not have a solid presence on the market, and thus most apparent limitation is the possible axis of movement as well as inability to perform turns using body movement. It is also cumbersome solution and as any other hardware solution will require additional space to store, and additional cost to purchase it, however since it only changes the HCI device it can be applied to any existing VR application(assuming compatibility between communicating protocols).

Redirected walking(RDW) - Implied by the name the technique relies on redirecting the user. An intermediate step that scales magnitude of the rotational and/or transnational input(Real path 2), to augment the path(Virtual path 2) of the user so it can fit a given area. Redirection algorithms can be carefully crafted, however it is increasingly popular to use machine learning in order to improve performance and also add additional inputs like eye-tracking for more elaborate implementations [4]. Redirection is possible because of the innate human difficulty to maintain a sense of direction when deprived of visual environmental clues [5]. However despite of the redirection human brain will still pick up the difference between perception and reality what is the direct cause of motion sickness, larger augmentation of the path will result in stronger discomfort.
Space required for infinite RDW is highly dependant on the technique and an individual user, therefore is measured at a point where a significant proportion of users doesn't feel an unacceptable level of nausea and disorientation. [3] suggests a 6x6 room to be an acceptable space for infinite walking, a 4x4 room proposed in [6] to be plausible, but not applicable to a wide audience, and a 44x44 room for RDW to be undetectable by most of the users, $36m^2, 16m^2$, and $1936m^2$ respectively. Safety does not differ from a usual VR setup, 0.5m on either side of the room for padding and indication if the user is leaving the dedicated area. This solution is not restricted in the direction of movement. The cost of RDW is additional processing power required to compute an optimal redirection path in real time. It also increases the entry barrier for users that are susceptible to motion sickness due to nature of difference in perception and reality discussed earlier [5].

Non-euclidean geometry(Portals) - The term "non-Euclidean geometry" generally refers to the study of geometries that are based on the negation of one or more of the Euclidean postulates. Otherwise it broadly generalises computer graphics exploits that are generally impossible in a day to day life. What is refereed to as *portal* in particular is the concept of two planes at any point of space that are linked to each other, so looking into one of the planes will present a view form the second one, same would happen with locomotion walking into one would appear as walking out of the second one 4, act of such locomotion is called *teleportation*, ex. a door from home to the work place and no traffic, what a great idea! Although not possible in the real(Euclidean) world, as it would mean travel with infinite speed or being in two position at the same point in time, there no physical law that prevents such behavior to be generated using computer graphics, where similar techniques are already used to create efficient reflections, shadows etc. These "doors", *portal* can be arranged in a way where an infinite amount of rooms can be overlapped(Navigable virtual space 3). [7] can be traversed without exiting a the dedicated area in the real world. This solution is especially potent in gaming where it can be used a main game-mechanics feature [8].
Theoretically there is no minimal space required to use this solution as it relies on overlapping the space available, however for any significant locomotion there needs to be at least space of a classical setup roughly $10m^2$. Similarly to RDW safety is no different to a usual VR setup. As limitation the largest single virtual room from the once being traversed can be no larger than the available space. This solution can excel in specific scenarios, especially closed spaces like VR museums or exhibitions, puzzle games, training ground, etc. The cost , however the cost is that the solution can not be applied to an arbitrary project and has to be built around it.

4.4. Assessment

This study has provided 3 solutions that are currently viable that are in active development. However as the goal is to bring it to the mass consumer, and they are yet to reach that point and are likely to remain in development or be a niche in the nearest future. Active repositioning although already present on the market the cost of a safe an precise hardware remains high. RDW is very promising as it does not require additional hardware and acts as a post-processing for the input, but still detracts consumers as increases disorientation, nausea in especially prolonged sessions. Non-eucledian geometry can already be successfully be implemented, however would only be applicable to certain edge cases described earlier.

5. Roomfininty - The Biggest VR room in The Smallest amount of physical space

5.1. Requirements

5.2. Functional requirements

- **Navigation and Exploration**: The user must be able to navigate navigate the 3D environment in a similar manner to real world(walk and look around) by using a keyboard and a mouse/track-pad input.
- ***Portal* rendering and interaction**: The *portal* must be rendered correctly and the user must be able to interact with it according to the explanation provided in this paper.
- **Data import**: The engine must support import of 3D scenes of .gltf file type to the extent of it's use, according to the glTF2 specification [9]. Support of all the features in the specification is not required, only ones used for the deliverable.

5.3. Non-functional requirements

- **Runtime performance**: This refers to the performance of the program once it has finished loading. Program should achieve a above 30 frames/second(FPS) at runtime on a modern device and browser that supports WebGL2 and can render WebGL2 graphics.
- **Visually immersive**: Refers to visual bugs like flicker and inconsistencies towards the rules of the created space must be as minimal as possible or ideally completely eradicated

5.4. Design

The project takes form of a 3D engine and a User Interface(UI) that allows to interact with environments created using that engine. The project is written in a strongly object oriented manner where many features are encapsulated inside classes with methods operating their instances. The main language of the project is JavaScript(JS) that is used for entirety of the engine. Communication with the Graphical-Processing-Unit(GPU), which is required for 3D rendering is done through a graphics and compute API "WebGL2". A small portion of the engine is written in GLSL which has a computational role of the final picture that combines lighting, cameras, objects. etc. Web-pages and UI elements are created using HTML and CSS. And finally a small Python3 script is used to create a local server and launch the application.
The deliverable was split in the following development stages:

5.4.1. Boilerplate. This stage refers to the early development of the the engine where time was taken to brainstorm and develop the architecture of the engine, as well as to create a prototype for the project 13, 14. This is a very important step in any development as it allows for a better evaluation of crucial decisions to insure that the project plausible to complete given the constraints.

5.4.2. Engine. At this stage the material created in boilerplate stage is finalised. And the flowing functionality is produced:

- Data import - In the beginning of the project the file type chosen for import was .obj. .obj contains mesh data for objects such as vertex positions, normals, indices, and texture coordinates(UVs). The choice was made as the file type is simple and small to implement, perfect for early development and prototyping. However as the file type is made for storing single objects it becomes unmanageable to import and dynamically initialize large scenes.
  Therefore .gltf file type being a better choice for the final engine due to ability to export entire scenes with objects, cameras, material, lights, etc. was implemented be following glTF2 specification [9]. The implemented features are meshes with single primitives, perspective cameras, point lights, scene objects, basic PBR Metallic-Roughness materials.
- Player Controller - The choice for player controller is simple as classic first person controller(FPC) behaves like reality where the camera is the view and locomotion left, right, forward and back is controlled with "wasd" keys.
- Lighting - Lighting choice is more difficult when implementing *portal* planes as the light must path through them.
  Path tracing would be a great solution as it would be the most simple conceptually, light path that hits a portal plane will exit from the linked plane and continue the travel. However this comes at a high performance cost, significantly increasing complexity due to the need of optimisation.
  Another popular choice when implementing portals is to remove lighting entirely in which case in order for shapes not be flat 12, 11, every edge is drawn with a distinct color. However this reduces the use case scenario and limits the aesthetic choice of the project.
  Therefore a compromise was to implement a physically based rendering(PBR) with point lights but leave shadow out, to approximate light teleportation only the view angle is needs to be changed in the light equation.
- Computational classes for matrices, vectors, etc. - The only note is that 4x4 matrices are column-major.

The final data structure is following 9. The engine class stores an active scene that contains dictionaries of "data-objects" which are collections of data and are not represented on the scene and "scene-objects" that all inherit from "Empty Object" with a "transform" that allows them to have a world position represented on the scene.
The simplified engine execution cycle is following 10. When the web-page is loaded scene and object data is loaded, the engine is initialized and starter. It then enters what is called a "Render Loop", a state that is responsible for any recurrently updating functions, in this case canvas is refreshed following by the scene is drawn. Before a next loop cycle an "Update" event is dispatched, to which other functionalities lime move in FPC can be subscribed to.

5.4.3. Portal. This is the main stage in the production as it focuses on creation of portals and HCI.
The principle of the working of the portal is visualised on the fig. 5. The blue dot represents the user, a triangular shape is the view of the user, 2 black boxes are 2 linked portals , a secondary camera(orange dot) with its own view is positioned equally relative to the portal 2 as player is positioned to the portal 1. For portal to work the colored part of the portal camera is drawn onto the canvas of portal 1.
The two popular options for rendering portals is to use stencil buffers or to use frame buffers. Both methods rely on rendering the scene a second time from a second view and then masking out objects that are on the inside of the the portal plane. In case of a frame buffer this is done through use of textures and in case of stencil buffer the mask is stored directly in the GPU. Masking refers to omission of a part of the picture depending on a certain condition, in case of the stencil test 6. The choice for this project is to use stencil buffer, mainly as an experiment as both methods are certainly viable. The goal here is to create a *portal* that is *immersive*, see Requirements. A further explanation of the workings of a portal is present in the production.

5.4.4. Demos, Landing page & Polish. At this point of the development most of the code is written and the majority and the rest of the resources are used to produce the following:

- Launcher - a small python script that launches the whole program in a local-host environment with required header such as ”Access-Control-Allow-Origin” to allow data not embedded into the HTML pages to be loaded.
- Landing page - this page contains a logo and a scrollable list of select-able demos with a title and a descriptive image, mock-up: 13.
- Demos - 3D environments that can be created in any 3D modeling software, in case of this project Blender.

5.5. Production

This section contains a detailed explanation for the functionality described in the requirements section of the report: Portal rendering and interaction. Some of the more basic concepts mentioned as pre-requisites are omitted from the explanation.

5.5.1. Portal rendering and interaction. The feature is handled by the portalDraw function that belongs to the SceneGraph class 1. The function is repeatedly called from the Engine class on each frame.
Before further explanation of rendering stencil buffer mentioned earlier is an array of values on for each pixel of the screen simmular to the depth buffer, that can be tested against reference values, this is called a stencil-test. WebGL provides following function to operate it:

- stencilOp(fail, zfail, zpass) - This function defines what operation(increment, decrement, keep, etc) on the stencil value is preformed on a per-pixel basis when the stencil-test fail(fail), when the stencil-test passes(zfail) but depth-test fail and when both tests pass(zpass).
- stencilFunc(func, ref, mask) - Describes the stencil-test itself, ”func” - the comparison operation (more, less, equal, never, etc.), ”ref” - the reference value to test against, ”mask” - bit-wise value that is used for the bit-wise AND between the reference value and the stencil value after the stencil test
- stencilMask(mask) - layer of stencil values that is masked

Also the following functions were developed for the engine before attempting rendering:

- relativeMirror(from, to)2 - the function calculates a transformation matrix that describes a mirror position of a camera from a ”from” position relative to the ”to” position. This is used to compute the position of the portal camera compared to the player camera(5) and teleportation.
- clipProjectionMatrix(clipPlane)4 - the function that adjust the near-clipping plane of the camera frustum to match a plane ”clipPlane” . This function is used to prevent any objects that might appear in front of the portal camera from abstracting the view.
- boundingBox()3 - Function calculates an Axis-aligned bounding boxe for a mesh. Used to detect collision.
- boundingRadius(axisFilter) - Function calculates a largest radius that can bound a bounding box. ”axisFilter” - selectively omits one of the axis of the bounding box. Used to check whether the player is interacting with the portal

With the concept of Non-euclidean portal and stencil buffers explained, plan for rendering the portals is following:

- **Lines 24-34:** Set-up a negative-test to generate reference stencil values by drawing the portal object exclusively to the stencil buffer. This is done by disabling the color and the depth buffer and enabling the stencil test. Stencil operation for failed test is increment to record the stencil values, stencil function is never, meaning that the stencil will never pass(negative-test).
- **Lines 36-52:** Draw all of the objects on the scene from the portal camera(5) which is computed using the relativeMirror function. And adjust the frustum using clipProjectionMatrix function.
- **Lines 55-64:** Restart the stencil test using decrement stencil operation, this will separate the drawn stencil from the ones drawn later.

The following image is produced when only the stencil part is drawn 7. Once all of the portals are rendered the camera setup is reset and the planes are written into the depth buffer on the scene **Lines 67-73**. And finally all of the scene objects are drawn as regular 8.
With functions defined earlier teleportation is rather simple **Lines 6-20**. Before rendering calculate on which side of the portal the user is(dot product of difference in positions). If the side is different to the one on the previous render, and

the player is in contact with the portal plane(checked using boundingRadius) teleport the player to the relative mirror position.

5.6. Assessment

The deliverable successfully achieved the all of the functional requirements for the project. And was efficient enough to run above the required frame rate on multiple tested devices(mac-book air and hp omen laptops).
However the Visual immersion was hampered by a rare visual bug which is produced when the player camera is that close to the portal plane that the view momentarily clips through the portal plane.

Acknowledgment

I would like thank every rock that happened to be on my way for just barely not breaking my legs.

6. Conclusion

This paper has gone over 3 physical VR navigation technique identifying their key features and providing a detailed explanation for them after explaining the problem space in VR.
The deliverable has also produced a 3D WebGL engine that focused around rendering a non-euclidean concept portals. The process of rendering was explained in detail.

7. Plagiarism statement

I declare that I am aware of the following facts:

- As a student at the University of Luxembourg I must respect the rules of intellectual honesty, in particular not to resort to plagiarism, fraud or any other method that is illegal or contrary to scientific integrity.
- My report will be checked for plagiarism and if the plagiarism check is positive, an internal procedure will be started by my tutor. I am advised to request a precheck by my tutor to avoid any issue.
- As declared in the assessment procedure of the University of Luxembourg, plagiarism is committed whenever the source of information used in an assignment, research report, paper or otherwise published/circulated piece of work is not properly acknowledged. In other words, plagiarism is the passing off as one's own the words, ideas or work of another person, without attribution to the author. The omission of such proper acknowledgement amounts to claiming authorship for the work of another person. Plagiarism is committed regardless of the language of the original work used. Plagiarism can be deliberate or accidental. Instances of plagiarism include, but are not limited to:
  1) Not putting quotation marks around a quote from another person's work
  2) Pretending to paraphrase while in fact quoting
  3) Citing incorrectly or incompletely
  4) Failing to cite the source of a quoted or paraphrased work
  5) Copying/reproducing sections of another person's work without acknowledging the source
  6) Paraphrasing another person's work without acknowledging the source
  7) Having another person write/author a work for oneself and submitting/publishing it (with permission, with or without compensation) in one's own name ('ghost-writing')
  8) Using another person's unpublished work without attribution and permission ('stealing')
  9) Presenting a piece of work as one's own that contains a high proportion of quoted/copied or paraphrased text (images, graphs, etc.), even if adequately referenced

  Auto- or self-plagiarism, that is the reproduction of (portions of a) text previously written by the author without citing that text, i.e. passing previously authored text as new, may be regarded as fraud if deemed sufficiently severe.

References

[1] H. Iwata, "Walking about virtual environments on an infinite floor," in *Proceedings IEEE Virtual Reality (Cat. No. 99CB36316)*, 1999, pp. 286–293. [Online]. Available: https://doi.org/10.1109/VR.1999.756964

[2] N. C. Nilsson, S. S. andFrank Steinicke, and R. Nordahl, "Natural walking in virtual reality: A review," *Computers in Entertainment*, vol. 16, no. 8, pp. 1–22, 2018. [Online]. Available: https://doi.org/10.1145/3180658

[3] M. Rietzler, J. Gugenheimer, T. Hirzle, M. Deubzer, E. Langbehn, and E. Rukzio, "Rethinking redirected walking: On the use of curvature gains beyond perceptual limitations and revisiting bending gains," in *2018 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, 2018, pp. 115–122. [Online]. Available: https://doi.org/10.1109/ISMAR.2018.00041

[4] Y. Joshi and C. Poullis, "Saccadenet: Towards real-time saccade prediction for virtual reality infinite walking," *arXiv preprint arXiv:2205.15846*, 2022. [Online]. Available: https://doi.org/10.48550/arXiv.2205.15846

[5] J. L. Souman, I. Frissen, M. N. Sreenivasa, and M. O. Ernst, "Walking straight into circles," *Current Biology*, vol. 19, no. 18, pp. 1538–1542, 2009. [Online]. Available: https://doi.org/10.1016/j.cub.2009.07.053

[6] E. Langbehn, P. Lubos, G. Bruder, and F. Steinicke, "Bending the curve: Sensitivity to bending of curved paths and application in room-scale vr," *IEEE Transactions on Visualization and Computer Graphics*, vol. 23, no. 4, pp. 1389–1398, 2017. [Online]. Available: https://doi.org/10.1109/TVCG.2017.2657220

[7] R. Epplée and E. Langbehn, "Overlapping architecture: Implementation of impossible spaces in virtual reality games," 2021. [Online]. Available: https://doi.org/10.25972/OPUS-24577

[8] H.-J. Backe, "The aesthetics of non-euclidean game spaces," in *Game — World — Architectonics: Transdisciplinary Approaches on Structures and Mechanics, Levels and Spaces, Aesthetics and Perception*, M. Bonner, Ed. Heidelberg: Heidelberg University Publishing, 2021, pp. 153–167. [Online]. Available: https://doi.org/10.17885/heiup.752.c10385

[9] The Khronos® 3D Formats Working Group. (2021) gltf™ 2.0 specification. [Online]. Available: https://registry.khronos.org/glTF/specs/2.0/glTF-2.0.html#foreword

[10] "[BiCS(2021)] Bachelor in Computer Science: BiCS Semester Projects Reference Document. Technical report, University of Luxembourg (2021)."

[11] “[BiCS(2021)] BiCS Bachelor Semester Project Report Template. https://github.com/nicolasguelfi/lu.uni.course.bics.global University of Luxembourg, BiCS - Bachelor in Computer Science (2021).”

8. Appendix

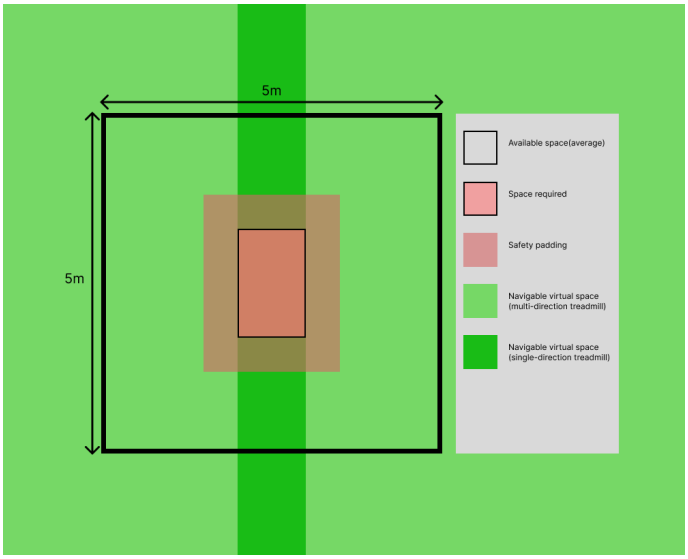


Fig. 1: Treadmill

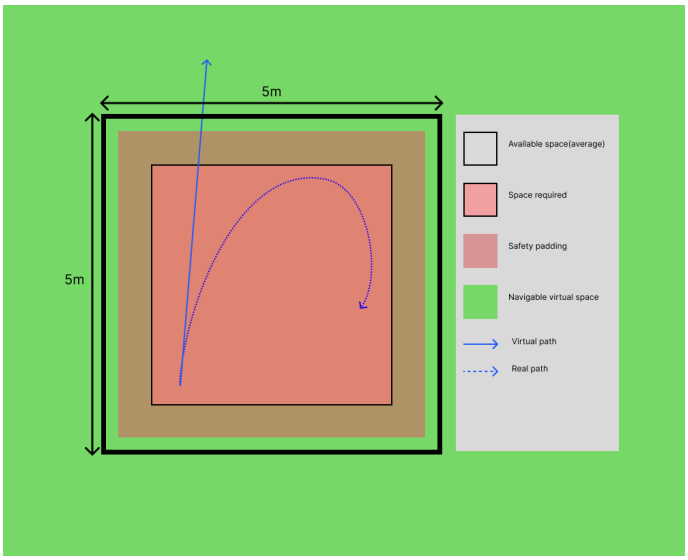


Fig. 2: RDW

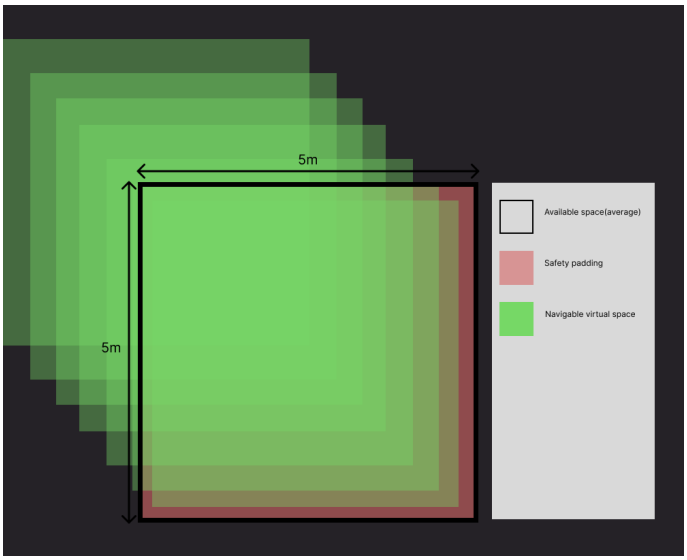


Fig. 3: Non-eucledian space(Portal)

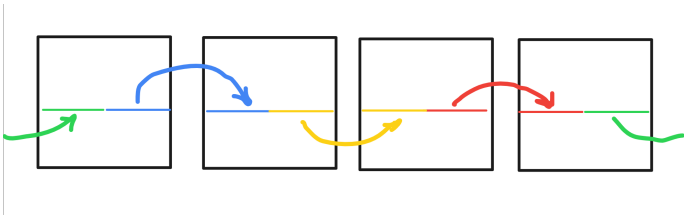


Fig. 4: Infinite locomotion through teleportation

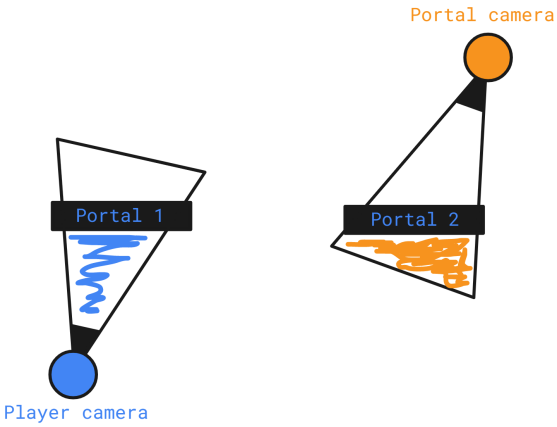


Fig. 5: Portal Cameras

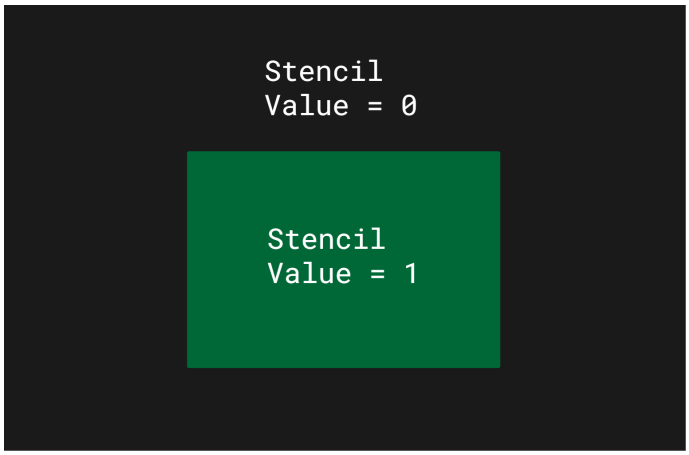


Fig. 6: Stencil Test

8.1. Source Code

```
portalDraw() {
    //for all portals evaluating on which side of
    the portal an observer is
    const newSides = this.portalLinks.map(link =>
    Vector3.side(link.master.transform.position,
    this.mainCamera.transform.position, link.master.
    transform.back));

    //handling teleportation
    for(let i = 0; i < this.portalLinks.length; i++)
    {
        const {master, link, side} = this.
    portalLinks[i];
        const newSide = newSides[i];

        let collide = Vector3.distance(master.
    transform.position, this.mainCamera.transform.
    position) < master.mesh.boundingRadius(new
    Vector3(1, 0, 1));

        if(typeof side === "undefined"){
            this.portalLinks[i].side = newSide;
        } else if(newSide !== side && collide){
            this.mainCamera.relativeMirror(master.
    worldMatrix, link.worldMatrix).inverse.
    toTransform(this.mainCamera.transform);
            this.portalLinks = this.portalLinks.map(
    link => {link.side = Vector3.side(link.master.
    transform.position, this.mainCamera.transform.
    position, link.master.transform.back); return
    link});
        }
        this.portalLinks[i].side = newSide;
    }

    //drawing the insides of the portal(the stencil
    part)
    for (const {master, link, side} of this.
    portalLinks) {
        //drawing frame into stencil
        gl.colorMask(false, false, false, false);
        gl.depthMask(false);
        gl.disable(gl.DEPTH_TEST);

        gl.enable(gl.STENCIL_TEST);
        gl.stencilOp(gl.INCR, gl.KEEP, gl.KEEP);
        gl.stencilFunc(gl.NEVER, 1, 0xff);
        gl.stencilMask(0xff);

        master.drawObject();

        //drawing scene inside of the portal
        gl.colorMask(true,true,true, true);
        gl.depthMask(true);

        gl.enable(gl.DEPTH_TEST);

        gl.stencilMask(0x00);
        gl.stencilFunc(gl.EQUAL, 1, 0xff);
```

Fig. 7: Scene inside the of portal

Fig. 8: Final Picture

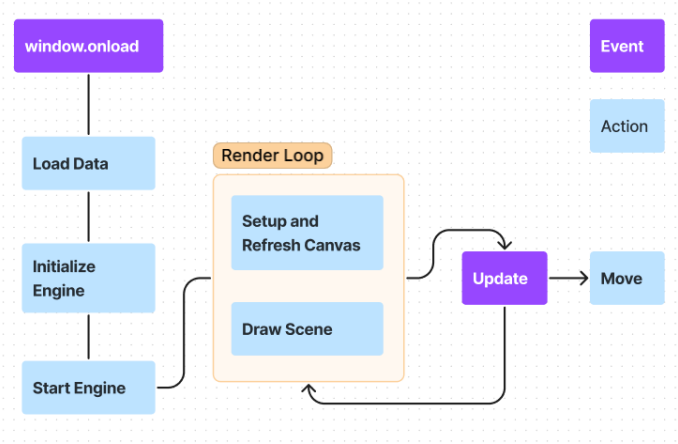


Fig. 10: Simplified engine execution cycle

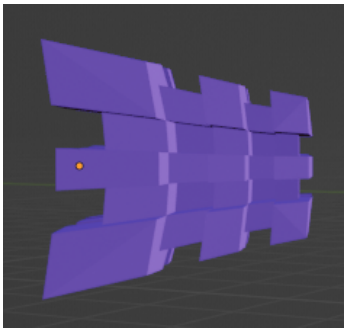


Fig. 11: With-lighting example

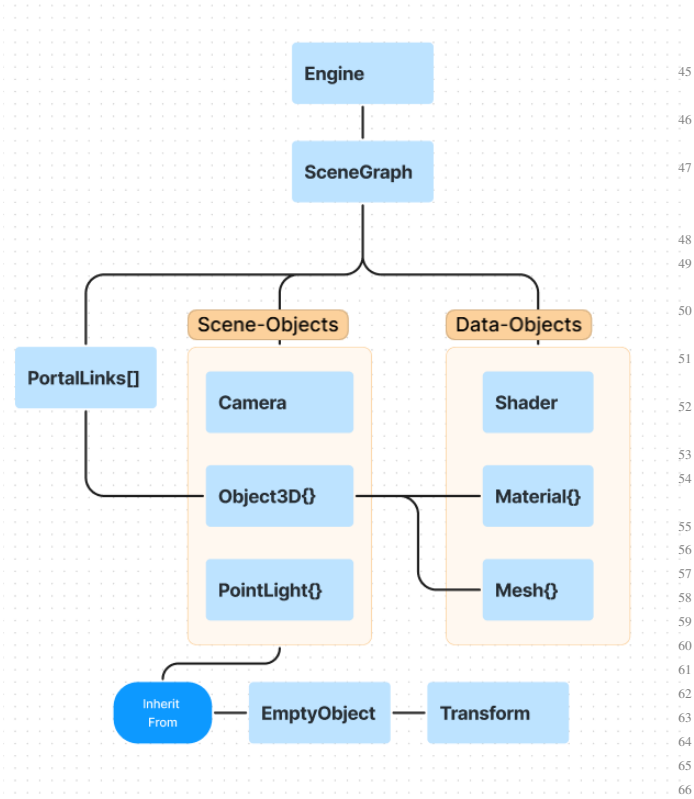


Fig. 9: Engine Data Structure

```
    let rm = this.mainCamera.relativeMirror(master.worldMatrix, link.worldMatrix);
    let proj = this.mainCamera.projectionMatrix.copy;
    proj.clipProjectionMatrix(Camera.clippingPlane(master.transform, this.mainCamera.viewMat, side));

    gl.uniformMatrix4fv(this.shader.uniforms.projection, false, proj.glify);
    gl.uniformMatrix4fv(this.shader.uniforms.view, false, rm.glify);
    gl.uniform3f(this.shader.uniforms.cameraPosition, ...rm.inverse.position);
    this.defaultDraw([Object3D.objectTags.Portal]);

    //resetting stencil test to separate stencils of every portal
    this.mainCamera.updateUniforms(this.shader);

    gl.colorMask(false, false, false, false);
    gl.depthMask(false);
    gl.enable(gl.STENCIL_TEST);
    gl.stencilMask(0xff);

    gl.stencilFunc(gl.NOTEQUAL, 1, 0xFF);
    gl.stencilOp(gl.DECR, gl.KEEP, gl.KEEP);
    master.drawObject();
}

//drawing portals to depth buffer from original point of view
gl.disable(gl.STENCIL_TEST);
```

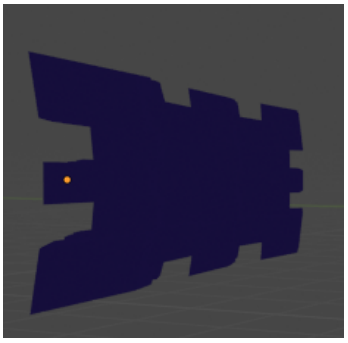


Fig. 12: Without-lighting example

```
    gl.colorMask(false, false, false, false);
    gl.depthMask(true);
    gl.clear(gl.DEPTH_BUFFER_BIT);

    this.portalLinks.forEach(link => link.master.drawObject());

    //drawing the rest of the scene with default view and setting
    gl.colorMask(true, true, true,true);
    this.defaultDraw([Object3D.objectTags.Portal]);
}
```

Listing 1: portalDraw() function

```
relativeMirror(from, to){
    return this.viewMat.multiply(from).multiply(Matrix4.identity().rotateY(glMath.toRad(180))).multiply(to.inverse);
}
```

Listing 2: relativeMirror() function

```
get boundingBox(){
    function leftFillNum(num, targetLength) {
        return num.toString(2).padStart(targetLength, 0).split('');
    }

    function everyNthElement(arr, n, offset) {
        return arr.filter((_, index) => (index - offset) % n === 0);
    }


    const sortedCoordinates = [0, 1, 2].map(offset => everyNthElement(this.#POSITION, 3, offset));
    const minMax = sortedCoordinates.map(arr => ([Math.min(...arr), Math.max(...arr)]));
    return Array.from({length:8}, (_,i) => new Vector3(...minMax.map((item, index) => item[leftFillNum(i, 3)[index]])));
}

boundingRadius(axisFilter = Vector3.one()){
    return Math.max(...this.boundingBox.map(point => point.multiply(axisFilter)).map((point, index, arr) => index + 1 === arr.length ? 0 : Vector3.distance(point, arr[index + 1]))) * 0.5;
}
```

Listing 3: boundingRadius() and boundingBox() function

```
clipProjectionMatrix(clipPlane)
{
    const vcamera = new Vector4(
        (Math.sign(clipPlane.x) - this.entries[8]) / this.entries[0],
        (Math.sign(clipPlane.y) - this.entries[9]) / this.entries[5],
        1,
        (this.entries[10] / this.entries[14])
    )
    const n = -1 / Vector4.dot(clipPlane, vcamera);

    this.entries[2] = n * clipPlane.x;
    this.entries[6] = n * clipPlane.y;
    this.entries[10] = n * clipPlane.z + 1;
    this.entries[14] = n * clipPlane.w;
}
```

Listing 4: clipProjectionMatrix() function

8.2. Requirements

Operation: Demo selecting.

- Users: End user on the landing page.
- Description: The user shall be able to scroll through a collection of existing demos and select the one he wants to run.
- Parameters: List with available demos.
- Pre-condition: user is on the landing page.
- Post-condition: The user was correctly redirected to the desired demo page.
- Trigger: Scroll bar and “run demo” button.

Operation: Exploring the demo.

- Users: End user inside a demo.
- Description: The user shall be able to walk and pan his view around the demo using keyboard and mouse input.
- Parameters: None.
- Pre-condition: User is on the demo page.
- Post-condition: The player’s actions correctly correspond to the intent of the user.
- Trigger: “wasd” keys on the keyboard and mouse.

Operation: Interacting with the “portal”.

- Users: End user inside a demo.
- Description: The user shall be able to walk through and look into the “portal”(the desired effect is that the interaction is seamless and does not appear any out of ordinary).
- Parameters: Player coordinates and camera rotation.
- Pre-condition: User is on the demo page.
- Post-condition: The user can see a correct view from the portal and when walked through is translated to the correct destination.
- Trigger: looking at or moving through the portal.

8.3. State of the Art

- [2] Article that broadly looks at current situation of interaction with VR.
- [8] Chapter exploring use of non-Euclidean geometry as part of level-design.

- [7] Discusses overlapping of virtual reality space to more efficiently use real world space.

8.4. Landing Page

- On the figure 13 the user will see a list he can scroll of demo's, that he can chose to run by clicking "Play demo" button.

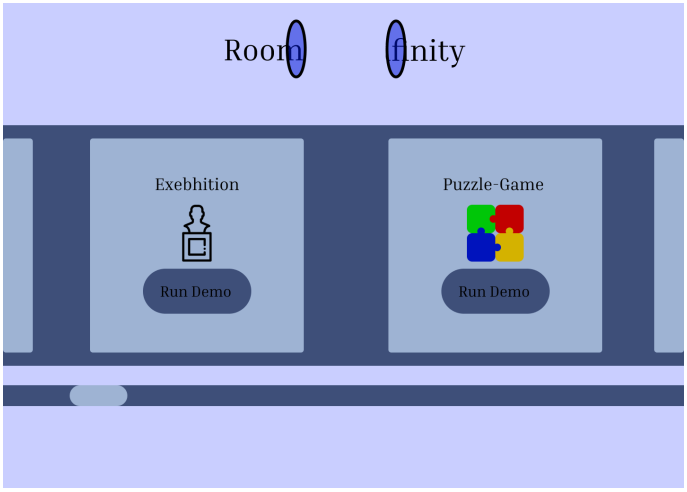
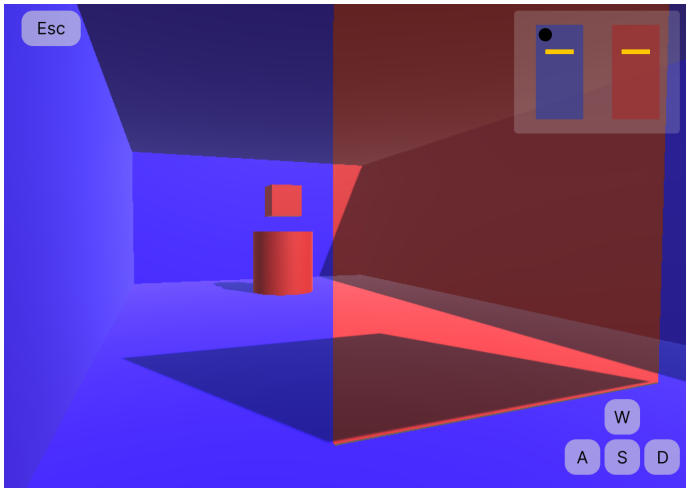


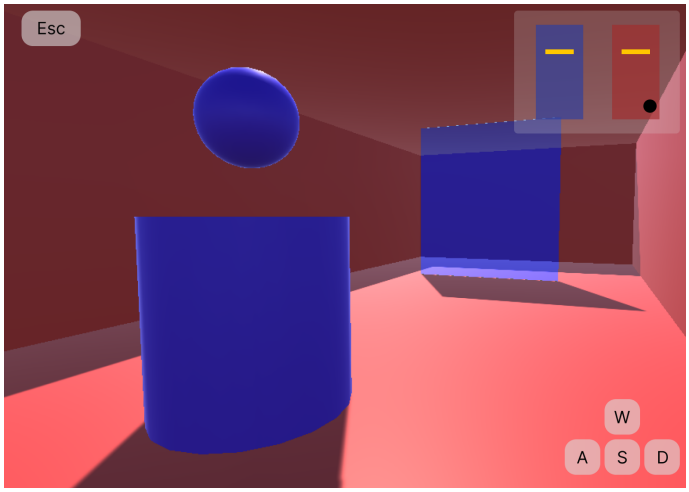
Fig. 13: Landing Page

8.5. Inside demo

- Inside the demo the user will control movement with "WASD" keys and pan around using mouse
- With the "Escape" button the user will go back to the landing page
- The box in the top right corner indicates position of the player with a black dot and position of portals with yellow bars



(a) Blue room



(b) Red room

Fig. 14: View from red rood and blue rooms