Designing Limitless Path in Virtual Reality Environment [⋆]

Raghav Mittal, Sai Anirudh Karre, and Y. Raghu Reddy

Software Engineering Research Center
IIIT Hyderabad, India
{raghav.mittal,saianirudh.karri}@research.iiit.ac.in
raghu.reddy@iiit.ac.in

Abstract. Walking in a Virtual Environment is a bounded task. It is challenging for a subject to navigate a large virtual environment designed in a limited physical space. External hardware support may be required to achieve such an act in a concise physical area without compromising navigation and virtual scene rendering quality. This paper proposes an algorithmic approach to let a subject navigate a limitless virtual environment within a limited physical space with no additional external hardware support apart from the the regular Head-Mounted-Device (HMD) itself. As part of our work, we developed a Virtual Art Gallery as a use-case to validate our algorithm. We conducted a simple user-study to gather feedback from the participants to evaluate the ease of locomotion of the application. The results showed that our algorithm could generate limitless paths of our use-case under predefined conditions and can be extended to other use-cases.

Keywords: Limitless Paths; Bounded Virtual Environment; Virtual Reality Products

1 Motivation

Virtual Reality (VR) environments are designed from a participant's perspective. In HMDs, participant is considered to be the center of the VR scene and the environment is oriented around the participant. This idea of centrality helps define the boundary of the VR environment for a given scene. VR practitioners normally design a full-scale scene to orient the participant within a fixed bounded environment rather than building a dynamic bounded environment due to various reasons. Some of the reasons like HMD limitations, physical space limits, dependency on additional hardware support, poor scene baking, poor frame rate, etc. influence the VR practitioners to build navigation controls through hand-held devices by letting the participant stay stationary. This contradicts the idea of creating realness in VR scene as the participant is forced to navigate the scene through hand-held controllers but in reality s(he) is stationary. To help address this issue, we developed an approach to let the participant navigate beyond the control of a hand-held device by taking into account the following two factors:

- let the participant physically navigate in the VR environment without the influence of external haptic hardware.
- let the participant navigate seamlessly with a limitless path in a limited virtual play area.

The above two factors can be addressed by considering the underlying technical aspects

- As part of the predefined constraints, physical environment and virtual environment ratio must be maintained
- An algorithm needs to be built to generate a limitless path with no obstruction for the participant to navigate in a virtual environment setup in a limited room space.
- VR environment assets need to be generated based on the path and orientation of the participant in the virtual environment
- VR environment should appear infinite to the participant. However, physically the participant will still be navigating in a limited predefined space.
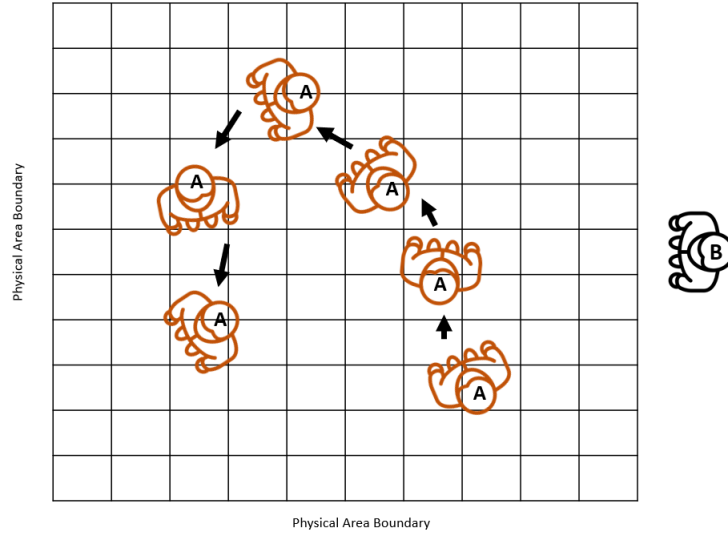
In this work, we use the term *'Limitless Path'* in the context of obstruction-free navigation in a VR environment within a limited physical environment. We define limitless path as a programmatically generated never ending path in a VR scene. This path is limitless and unbounded in terms of length. The progress of the path is automatic and the scene assets are generated inline with the generated path. For illustration consider Fig 1, person A - standing in a 10ft x 10ft physical grid wearing a HMD and another person B outside this physical grid. Person A loads a VR scene with the support of limitless path implementation and infinitely walks in this limited space. Person A experiences an endless walk in the VR scene. For person B, person A appears to be walking within a 10ft x 10ft physical grid area continuously in a random path within the limited physical area.

The rest of the paper is structured as follows: In section 2, we detail our approach towards designing and implementing path generation and boundary detection in a VR environment. In Section 3, we present the user study on a small set of participants. In section 4, we discuss the threats to validity. In section 5, we present some related work in this area and finally in section 6 we present some conclusions.
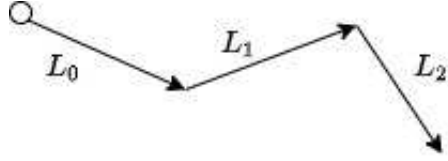
2   Our Approach

Continuous walking within the specified area (with in the scope of a VR scene or a physical area) is a critical aspect of path generation, as any generated path must be within the specified boundaries of the VR scene. Thus, path generation and boundary detection plays a key role in the progress of the participant along the limitless path. As part of our work, we designed and implemented both the path generation and boundary detection algorithms in a virtual environment.

Consider a use-case where a path has to be presented to the participant in a particular VR context. At the time of the VR scene generation, we are aware
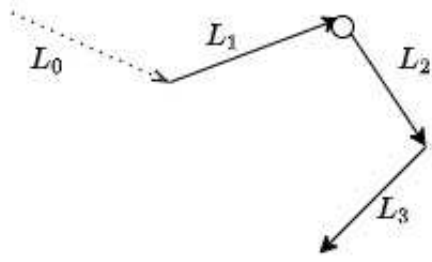
Fig. 1. Person A experiencing limitless path in a finite physical environment

Fig. 2. Path generated on start of system. L_0 starts from player's position.

of the start position of the participant with in a virtual environment. As the participant walks forward, the path is generated by addition and removal of line segments in the path to create a perception of continuity. This path generation system outputs the path as a line made of multiple connected line segments, as shown in Fig 2. The designed system will generate 3 successive line segments (shown in Fig. 2 as L_0,L_1, and L_2) from the starting point. When the player reaches the end of L_1, the first segment L_0 is removed, and a new segment L_3 is added to the end of L_2 as shown in Fig 3. This process keeps repeating until the participant terminates the program.

Each generated path line segment forms an obtuse angle with the previous line segment in most cases. This is done to ensure comfortable locomotion to the participant by avoiding sharp turns. It also reduces the possibility of the path intersection or path override when a new line segment is added to the path.

Fig. 3. L_0 is removed and L_3 is added when the player reached to the end of L_1.

2.1 Palling - Path Generation

In this section, we provide step-by-step details of our path generation algorithm. We term *'Palling'* as a user action to move forward in the virtual environment for ease of terms. *1 Pal* unit is equal to 1 unit distance taken by the user in the virtual environment. For purposes of simplicity, we assume the dimension of the rectangular bounded area from $(0,0)$ to (x,z). If the user starts at $0,0)$, the user can take z *Pal* units to reach $0,z)$ and x *Pal* units to reach $x,0)$. In order to generate a path, below are the necessary inputs:

- Player's starting position $_0$and head-yaw $\beta_0$at that position.
- Dimensions of the boundary (x,z), here x and z are the limits of the boundary in a plane along *x-axis* and *z-axis*.
- Path properties include segment length and path width i.e the perpendicular distance between parallel boundaries representing the width of the path in virtual environment.
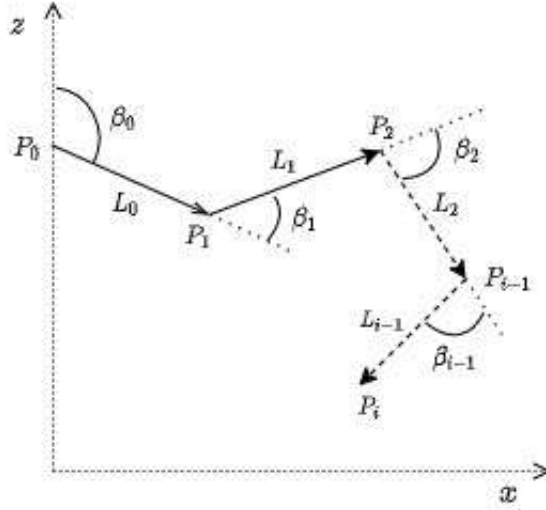
As shown in Fig 4, upon start of the application, the first line segment L_0 has a starting point P_0 and terminal point P_1 i.e. it is represented as $L_0 = \overline{P_0 P_1}$. Similarly, for L_1, P_1 and P_2 are starting and terminal points respectively. Eq 1 generalizes this for L_i. Currently the length is considered to be a constant value and provided as a necessary input. We plan on computing this automatically depending on the boundary area in future.

$$L_i = \overline{P_i P_{i+1}} \tag{1}$$

In the given play area, the coordinates of position P_{i+1} are calculated as shown in following equations 2 and 3, where $0 <= i <$ total number of segments generated since the start of the application. Here β_0 is head-yaw (HMD's orientation along y-axis in virtual environment) of the user.

$$P_{i+1}(x) = l * \sin \beta_i + P_i(x) \tag{2}$$

$$P_{i+1}(z) = l * \cos \beta_i + P_i(z) \tag{3}$$

Fig. 4. Locomotion of a participant in the play area

When we recursively run equations 2 and 3, we generate a limitless path in a defined boundary of $D(x, z)$. The generated path is instantaneous, nonlinear, and limited by certain Pal units due to the bounded area. The upcoming P_i needs to be generated by detecting the proximity of P_{i-1} to the boundary. When P_{i-1} is not close to the boundary, β_i is set to a random value in range $\{\beta_{i-1} - \pi/2, \beta_{i-1} + \pi/2\}$. The procedure of generating the next line segment L_i for a given position P_i at a boundary is defined on β_i value.
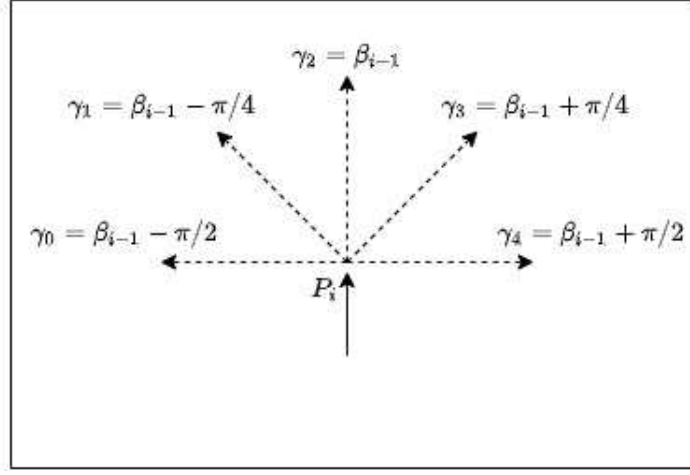
2.2   Pragging - Boundary Detection

In this section, we detail our boundary detection algorithm. For ease of terms, we refer to a user instance on detecting a boundary or hitting a boundary as *'Pragging'. 1 Prag* unit is equal to one hit at a boundary.

To ensure that the player doesn't cross the boundary of the given application, the generated path must not span beyond the bounded-area. To achieve this, the bounded-area is enclosed into wall-like colliders. We define *'j'* as a value that equally divides a 180° range into multiple possible rays as shown in Fig 5. Whenever a new point P_{i+1} has to be generated, *j+1* number of rays are projected in multiple directions with certain angles called as γ where k is in range $\{0, j\}$ and $j > 0$. The range of angle here is $\beta_{i-1} - \pi/2, \beta_{i-1} + \pi/2$
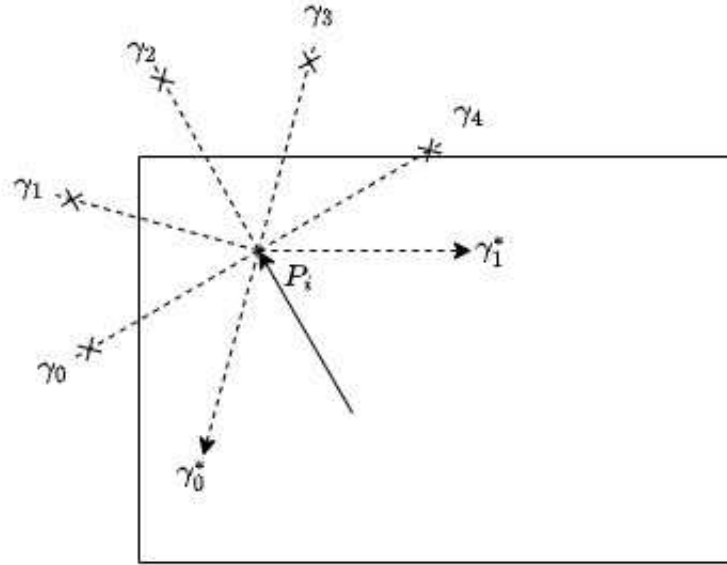
$$\gamma_j = \beta_{i-1} - \pi/2 + ((\pi/j) * k) \tag{4}$$

If the value of j=4, we have j+1 rays i.e. 5 rays at equal angles between $\beta_{i-1} - \pi/2, \beta_{i-1} + \pi/2$ are called $\gamma_0, \gamma_1, \gamma_2, \gamma_3, \gamma_4$ as shown in Fig 5. Here the source of the rays is P_i and length is equal to *path_length + path_width/2*
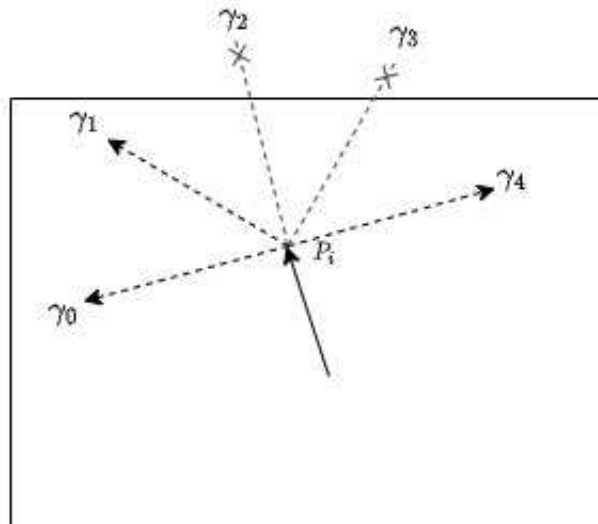
Fig. 5. Dashed lines represent rays, solid line represents the direction of the line segment L_{i-1} of the existing path. Here $j = 4$.

If a ray collides with the play area boundary, we term it as 1 Prag unit. If $j=a$, we have $a+1$ rays generated. Out of these $a+1$ rays, one of the γ_i direction is chosen to generate a path L_i. Below are the possible cases in which the generated rays can collide with a boundary:

- If $\gamma = a+1$ *Prag* units, i.e. all generated rays hit the boundary as shown in Fig 6. This means that the P_i is at the corner of the bounded area. In this case, $\pm 135°$ is the way to go away from the boundary. Two more rays are shot in directions $\gamma_0^* = \beta_{i-1} - 3\pi/4$ and $\gamma_1^* = \beta_{i-1} + 3\pi/4$ to avoid corner as an escape strategy. The ray which don't hit the boundary is chosen as the direction for L_i. If P_i is equidistant from boundaries then one of the ray from γ_0^* and γ_1^* is chosen randomly.

- If $1 \leq \gamma \leq a$ *Prag units* i.e. not all but atleast one ray hits the boundary as shown in Fig 7 and anyone of the non-hitting rays can be used to generate L_i.

- If $\gamma = 0$ *Prag* units, i.e. none of the rays hit any of the boundaries. Here the path is free from boundaries as shown in Fig 5. Then the β_i is randomly chosen from range $\beta_{i-1} - \pi/2, \beta_{i-1} + \pi/2$ for generating the upcoming L_i.

Fig. 6. When all the rays $\gamma_0....\gamma_{j+1}$ hit the boundaries then two more rays are generated to decide the direction of L_i.

Fig. 7. The γ for next L_i will be chosen from γ_0, γ_1, γ_4.

2.3   PragPal Algorithm

As part of our work, we conduct pragging and palling simultaneously to generate a limitless path for a user in a virtual environment. Algorithm 1 provides the pseudo-code of our concept, which is used in the simulation [1].

Algorithm 1: PragPal Algorithm Overview

Input: user position, head yaw, line segment length, path width
Output: List of 2D points x,z which represent a path

1 **Function** `GeneratePoint_Pal`(*beta, point*)**:**
2 $point.x = segment\_length * \sin(beta) + point.x$
3 $point.z = segment\_length * \cos(beta) + point.z$
4   **return** point
5 **Function** `GenerateBeta_Prag`(*beta, point*)**:**
6   j = 4 // can be any value greater than 1
7   valid_directions = []
8   for k in range (0, j+1): // k = 0, 1, 2, 3, 4
9   ray_direction = $beta - \pi/2 + (\pi/j) * k)$
10   ray = Generate ray in direction ray_direction from *point*
11   if ray do not hit boundary:
12   valid_directions.push(ray_direction)
13   if valid_directions.size<0:
14   beta = randomly pick element from valid_directions
15   else:
16 $beta$ = randomly choose from range $\{beta - \pi/2, beta + \pi/2\}$
17   **return** beta
18 **Function** `Main`**:**
19   point = current position of player: [x, z]
20   beta = current head-yaw of player in radians
21   points_list = [] // points represent the shape of current path
22   points_list.Append(point)
23   **OnSceneStart:** // called only on initialization of the scene
24   for i in range(4):
25   point = GeneratePoint_Pal(beta, point)
26   points_list.Append(point)
27   beta = GenerateBeta_Prag(beta, point)
28   **OnSceneUpdate:** // called whenever player reaches end of segment
29   points_list.pop() // removes first element
30   point = GeneratePoint_Pal(beta, point)
31   beta = GenerateBeta_Prag(beta, point)
32   points_list.Append(point)
33 **return**

---

[1] https://github.com/raghavmittal101/path_gen_sys/

2.4 Prototype Implementation of Virtual art gallery

To understand our work's effectiveness, we implemented the algorithm to build a VR based 'Virtual Art Gallery'. It is an endless corridor composed of two walls running parallel to each other. The user can walk through the gallery to explore the art items displayed on these walls. When the user progresses in the forward direction, i.e., palls forward, using the tracked Pal units, the path generation logic updates the upcoming path and auto-generates the corridor with relevant assets in the VR environment.
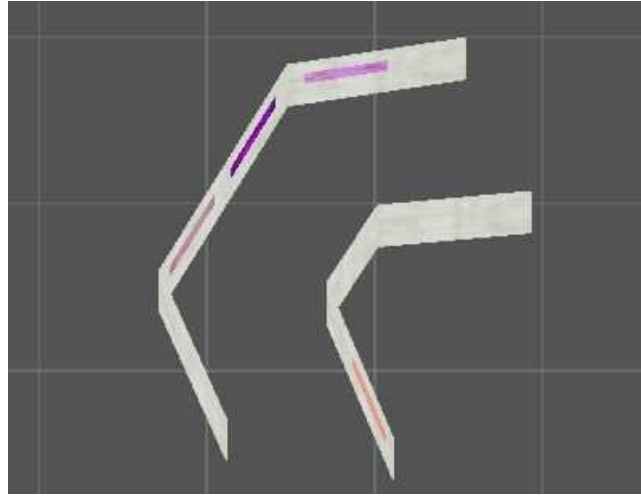
The corridor consists of two wall-like 2D mesh structures positioned parallel to the path. Whenever the new path is updated, the assets related to the corridor are also updated. In our example application, a set of royalty-free floral drawing images are displayed in the gallery. As the user navigates in the scene, Prag units are simultaneously tracked to detect boundaries to generate the possible paths. A collider trigger placed at the end of each line segment in the current path captures the Prag units. Whenever the user reaches a turn, this method is triggered. Thus generating an infinite limitless path for the participant in VR.

Fig. 8. Inside view the virtual art gallery.

We developed this use-case in Unity3D 2019 LTS, and tested it using Oculus Quest 2019 HMD. The images in the scene are placed at the player's standing height. This is to provide a comfortable view of the player during locomotion.

Dimensions of each image are set to 0.5 x 0.5 units in Unity3D (50 cm x 50 cm). A margin of 0.1 units was given on both sides of each image to keep the walls less cluttered as shown in 8. The number of images placed on each wall varied because all the walls cannot necessarily be of the same length, as shown in figure 9. Here, the maximum perpendicular distance between the walls called *path_width* is always maintained. This was set to 1.2 meter to have enough space for comfortable locomotion. The maximum length of a segment *segment_length* of path was set to 1.3 meter. Physical space availability is an important factor to consider while deciding the values of path_width and segment_length. A larger space can accommodate a broader and longer corridor.

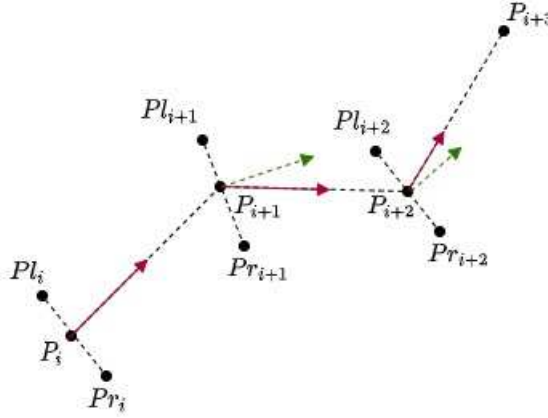**Fig. 9.** Top view of the virtual art gallery.

2.5   Corridor Generation

The Pragpal algorithm generates P_i, P_{i+1}, P_{i+2} ..... and so on. It outputs points that can form a line to be used by a participant to traverse the limitless path in a virtual environment. In order to generate a corridor for walking, as shown in Fig. 8 and Fig. 9, we generate points Pr_i and Pl_i on right and left side of each point P_i respectively. These two new points Pr_i and Pl_i are equidistant from P_i and the distance between P_i and Pr_i or Pl_i is half the path width (*path_width*). In the current version of our approach, the *path_width* is a static value defined by the programmer and is set as a default width of the visible or imaginary corridor. However, the path width may be changed depending the size of the boundary and future iterations of our work shall accomodate dynamic generation of path-width based on the VR scene and bounded area size. As P_i, P_{i+1} are generated by PragPal algorithm, the red arrow in Fig 10 from P_i is normal vector to

Pr_i and Pl_i towards P_{i+1}. Once we reach P_{i+1}, we move towards P_{i+2} along the red arrow. However, to position Pr_{i+1} and Pl_{i+1}, we generate a average vector (shown as green dotted arrow) of $\overrightarrow{P_{i+1} - P_i}$ vector and $\overrightarrow{P_{i+2} - P_{i+1}}$. The resultant average vector is dotted green arrow. Using this average vector, we generate Pr_{i+1} and Pl_{i+1}. Similarly, Pr_{i+2} and Pl_{i+2} and Pr_{i+3} and Pl_{i+3} so on will be created along with PragPal points. Joining all Pr_i's and Pl_i's points will provide us the corridor path required for our virtual art gallery. The code-implementation of this corridor generation[2] is included as part of our virtual art gallery scene. Using this corridor generation method, we generate boundary to the path width for better path visualization.

Fig. 10. Placement of left and right points for corridor generation.

This use-case is a simple instance of validating the limitless path approach. Our approach can be applied to various other use-cases or domains like entertainment, scientific research, studying spatial cognition, education, etc. to validate limitless paths.

3   User Experience Study

This section contains details about the user experience study conducted to understand our implementation's caveats, immersion, and ease-of-locomotion. Our use-case simulation is available on GitHub for practitioners for further experimentation [3] along with its visualization for practitioners' analysis [8].

Experiment Setup- We have set up a physical room with a scale of 24 feet x 17 feet at our university campus to run the virtual art gallery use-case. We used

---

[2] https://github.com/SebLague/Curve-Editor/tree/master/Episode%2006
[3] https://github.com/raghavmittal101/path_gen_sys/

Oculus Quest HMD, a standalone virtual reality headset with inside-out tracking capability. It helps determine the subject's position and orientation with no additional hardware apart from the HMD itself. Its technical specifications include a display resolution of 1440 x 1600 pixels per eye, 72Hz refresh rate, Qualcomm Snapdragon 835 processor, and 4GB RAM. As part of our VR scene, a set of 20 images are presented in the virtual art gallery. For test purposes, these images are set to be repeated till the termination of the environment. Due to the Coronavirus pandemic, we conducted our tests under COVID-19 protocol and limited the number of users who participated in the study. Special care was taken to sanitize the apparatus after each trial and follow. VR face-masks were provided to the participants to avoid direct face contact with the HMD headset.

Participants - All the subjects who participated in this study were university students/staff recruited randomly. Among the participants, there were six males and four females. The mean age was 24.5 years. Before undergoing the study, we asked them a few questions to understand their prior exposure to VR. We observed that 35% of the participants had used video/computer games regularly, and 29% had prior VR experience.

Tasks - After obtaining participant consent and educating them about VR induced sickness, we asked them to complete the following tasks in the given order:

– **Demographic Survey**[4]: Participants were asked fill out data regarding Age, Sex, Gaming experience, and VR experience.
– **Exploration Task**: Participants had to locomote in the virtual art gallery VR Scene for about 5 minutes at their will. Post 5 minutes, the scene was terminated externally by the experimenter by alerting the participant verbally.
– **Questionnaires**: Participants were requested to fill Simulator Sickness Questionnaire(SSQ) [4] to understand simulation sickness while navigating the algorithm generated path, Igroup Presence Questionnaire(IPQ) [1] to understand the participant presence experience in the scene generated by the algorithm.

Results and Observations - Table 1 provides the results of our survey study. We observe that the subject has a good understanding of spatial presence and sense of being in the virtual reality scene despite automatically generating the subject's path in the virtual environment. The subjects experienced some disorientation and minor Nausea levels while being part of our VR scene automated path generation. There is a reasonable amount of involvement of participant while navigating in the VR scene.

---

[4] https://forms.gle/CW5kAWjAz7oTsyb26

Table 1. Mean and Weighted % of User Experience of Virtual Art Gallery

Factors	Mean	%Weighted Response
Nausea	1.3	6.19%
Oculomotor	3.2	15.2%
Disorientation	2.8	13.3%
Sense of being	5.2	74.2%
Spatial Presence	25.9	74%
Involvement	18.7	66%
Experienced Realism	14.8	52%
Presence	64.6	65%

4   Threats to Validity

Internal Validity - We conducted a controlled experiment of our approach using a limited setup. We incrementally recorded the experiment's updates and conducted a minimal study in an optimal setup consisting of a rectangular physcial room. We reviewed the study design with fellow researchers and gathered feedback. Our physical experiment results match the results of simulation studies conducted through automated means. However, given the covid restrictions, the interactions were fairly limited and virtual in nature. A more realistic in-person review of the study design could have yielded better insights.

External Validity - We made every attempt to conduct a simplified study of a use-case among the few participants. Our results show that our approach can be easily extended to a large population. However, a serious user-study with a large sample size is required to understand the underlying challenges of our approach. Additionally, creating different types of rooms in a iterative manner could have provided a different set of results.

Construct Validity - We coined Palling and Pragging as units to determine the motion and halt of a user in the virtual environment for ease of terms. It helped us establish the distance traveled versus the number of times the user hit the boundary. Using these two new units, we could ascertain the user's course and progress in a virtual environment.

Conclusion Validity - In our study, we don't claim our approach as optimal compared to existing methods. There is a possibility of more effective ways of limitless path generation. A comparative study among all available methods is essential to determine the statistical significance of our approach over others.

5   Related Work

Williams et al. conducted experiments on understanding the differences between locomotion in virtual environments using Joystick and locomotion through phys-

ical turning [15]. Their work is oriented towards understanding the spatial limitations through the different means of locomotion only. They found that large physical spaces are required for comfortable locomotion. A small space limits the length of the generated path. Such a problem can be solved by scaling up the translational gain or developing alternate methods of locomotion. Darken et al. [2] are the first to develop an omnidirectional treadmill, and Souman et al. have extended it to a Cyberwalk Omnidirectional treadmill to enable infinite locomotion in a finite space [11]. This setup requires external haptic hardware support and may not provide the appropriate end user experience in the virtual environment for locomotion. Juyoung et al. have discussed the differences in treadmill based Walk-in-place methods, and Non-treadmill based Walk-in-place methods [5]. Griffin et al. compare the locomotion techniques, which require a hand-held device for locomotion. For instance, teleportation versus the hands-free techniques like walk-in-place [3]. They observed that hands-free techniques offer a higher presence than hand-busy techniques.

Sun et al. proposed a saccadic redirected walking technique, which worked based on eye-tracking. It allows infinite walking in a room-scale environment without the need for hardware like ODTs. However, one of the major downsides of it was that it required HMD with eye-tracker and high processing resources. [13]. Matsumoto et al. employed a visual-haptic approach to let the user walk on an unlimited virtual straight line. Their method required installing a circular ring-like structure around which the user was required to walk while feeling the boundary with one hand [7]. Such setup makes the overall HMD setup immobile, expensive and less adaptable to the masses. Vasylevska et al. proposed flexible spaces technique in which the environment consisted of 4 rooms with partially overlapping areas, but the overlap was not visible to the user. Such an environment creates a perception of limitlessness because the user cannot see beyond the wall of their current room [14]. Such an approach can be tweaked to provide multiple path options to the participants. Suma et al. extended this work with an argument of maximizing walking in a limited space. Their algorithm dynamically modifies the layout of the scene, which consists of two overlapping rooms. Their overlapping is dynamically modified throughout the game-play to create an illusion of infiniteness, but not a true space [12].

Rietzler et al. proposed a redirection approach in which the user walks in a radius of 1.5 meters. They tried to address the motion sickness caused due to the mismatch between vestibular and visual inputs. According to their work, when walking in a circular path of radius more than or equal to 1.5 meters, the users cannot differentiate between walking in a circular path, or a straight line [9]. This was a unique observation and can be used for future advancements on non-haptic-based walking in the VR environment. Conventionally, teleportation methods do not require the user to walk, and hence it is not good for presence. However, Liu et al. presented a teleportation method in which a portal to a location is generated when the user points towards a location to teleport. The user has to walk and step into the portal to teleport. Here the portals are defined

in a single room-scale boundary [6]. Thus providing limitless locomotion through teleportation.

Ruddle et al. compared the performance of participants in an object searching task done in 4 different conditions. They include search in the real-world, search in the virtual environment through a screen display, search in VE through an HMD with controllers and search in VE with real locomotion. They found the performance to be the most accurate in VE with real locomotion when compared with real-world performance [10]. They observed that the natural locomotion interface is the most suitable interface for locomotion in virtual environments because it provides translational and rotational body-based information. Most of these studies are either haptic based or illusion based to achieve a limitless path.

6   Conclusion

In this paper, we presented an approach to implement limitless path generation in Virtual environments using our PragPal Algorithm. We discussed path generation and boundary detection methods used as part of the algorithm. We implemented the algorithm on a Virtual Art Gallery use-case to understand the robustness through simulation. We also conducted a small user study to capture feedback from real-world participants. Our results and observations are promising and provide reasonable motivation to proceed with our planned work. We are currently working on a comparative study through a systematic literature review to understand the merits and demerits of all available locomotion methods in regards to limitlessness.

As part of future work, we plan to conduct a large scale use-study to understand presence, accommodation, and ease of use experiences from real-world participants to make our algorithm flexible for other implementations. We plan to tweak our algorithm to enable flexibility with the input parameters, for example, length of the line segment, boundary size, width of the path, etc.

In the current approach, only a single path is generated. The participant has to take that path and can only move in the forward direction. Given that the prior line segments are removed in the process of forward movement, retracing the steps is not possible. We plan on extending the algorithm to facilitate create of a multi-path environment. This will have its own set of challenges, especially from a rendering and overlaying perspective when the participant can move either forward in different directions or retrace their steps to a certain point. We also plan to introduce assets in the use cases that involve interaction between the participant and the asset. This will facilitate development of more use-cases towards realizing limitless path navigation for recreation, phobia studies, banking and other education related applications that involve interaction for task completion.

References

1. Igroup presence questionnaire, `http://www.igroup.org/pq/ipq/`, accessed: 2021-02-04
2. Darken, R.P., Cockayne, W.R., Carmein, D.: The omni-directional treadmill: a locomotion device for virtual worlds. In: Proceedings of the 10th annual ACM symposium on User interface software and technology. pp. 213–221 (1997)
3. Griffin, N.N., Liu, J., Folmer, E.: Evaluation of handsbusy vs handsfree virtual locomotion. In: Proceedings of the 2018 Annual Symposium on Computer-Human Interaction in Play. p. 211–219. CHI PLAY '18, Association for Computing Machinery, New York, NY, USA (2018). https://doi.org/10.1145/3242671.3242707, `https://doi.org/10.1145/3242671.3242707`
4. Kennedy, R.S., Lane, N.E., Berbaum, K.S., Lilienthal, M.G.: Simulator sickness questionnaire: An enhanced method for quantifying simulator sickness. The International Journal of Aviation Psychology **3**(3), 203–220 (1993). https://doi.org/10.1207/s15327108ijap0303_3, `https://doi.org/10.1207/s15327108ijap0303_3`
5. Lee, J., Hwang, J.I.: Walk-in-place navigation in vr. In: Proceedings of the 2019 ACM International Conference on Interactive Surfaces and Spaces. p. 427–430. ISS '19, Association for Computing Machinery, New York, NY, USA (2019). https://doi.org/10.1145/3343055.3361926, `https://doi.org/10.1145/3343055.3361926`
6. Liu, J., Parekh, H., Al-Zayer, M., Folmer, E.: Increasing walking in vr using redirected teleportation. In: Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology. p. 521–529. UIST '18, Association for Computing Machinery, New York, NY, USA (2018). https://doi.org/10.1145/3242587.3242601, `https://doi.org/10.1145/3242587.3242601`
7. Matsumoto, K., Narumi, T., Ban, Y., Yanase, Y., Tanikawa, T., Hirose, M.: Unlimited corridor: A visuo-haptic redirection system. In: The 17th International Conference on Virtual-Reality Continuum and Its Applications in Industry. VRCAI '19, Association for Computing Machinery, New York, NY, USA (2019). https://doi.org/10.1145/3359997.3365705, `https://doi.org/10.1145/3359997.3365705`
8. Mittal, R.: Limitless path project (2021), `https://serc.iiit.ac.in/resources/projects/limit`
9. Rietzler, M., Deubzer, M., Dreja, T., Rukzio, E.: Telewalk: Towards free and endless walking in room-scale virtual reality. In: Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems. p. 1–9. CHI '20, Association for Computing Machinery, New York, NY, USA (2020). https://doi.org/10.1145/3313831.3376821, `https://doi.org/10.1145/3313831.3376821`
10. Ruddle, R.A., Lessels, S.: The benefits of using a walking interface to navigate virtual environments. ACM Trans. Comput.-Hum. Interact. **16**(1) (Apr 2009). https://doi.org/10.1145/1502800.1502805, `https://doi.org/10.1145/1502800.1502805`
11. Souman, J.L., Giordano, P.R., Schwaiger, M., Frissen, I., Thümmel, T., Ulbrich, H., Luca, A.D., Bülthoff, H.H., Ernst, M.O.: Cyberwalk: Enabling unconstrained omnidirectional walking through virtual environments. ACM Trans. Appl. Percept. **8**(4) (Dec 2008). https://doi.org/10.1145/2043603.2043607, `https://doi.org/10.1145/2043603.2043607`

12. Suma, E.A., Lipps, Z., Finkelstein, S., Krum, D.M., Bolas, M.: Impossible spaces: Maximizing natural walking in virtual environments with self-overlapping architecture. IEEE Transactions on Visualization and Computer Graphics **18**(4), 555–564 (April 2012). https://doi.org/10.1109/TVCG.2012.47
13. Sun, Q., Patney, A., Wei, L.Y., Shapira, O., Lu, J., Asente, P., Zhu, S., Mcguire, M., Luebke, D., Kaufman, A.: Towards virtual reality infinite walking: Dynamic saccadic redirection. ACM Trans. Graph. **37**(4) (Jul 2018). https://doi.org/10.1145/3197517.3201294, `https://doi.org/10.1145/3197517.3201294`
14. Vasylevska, K., Kaufmann, H., Bolas, M., Suma Rosenberg, E.: Flexible spaces: Dynamic layout generation for infinite walking in virtual environments. In: IEEE Symposium on 3D User Interface 2013, 3DUI 2013 - Proceedings. pp. 39–42. IEEE Symposium on 3D User Interface 2013, 3DUI 2013 - Proceedings (Oct 2013). https://doi.org/10.1109/3DUI.2013.6550194, 8th IEEE International Symposium on 3D User Interfaces, 3DUI 2013 ; Conference date: 16-03-2013 Through 17-03-2013
15. Williams, B., Narasimham, G., McNamara, T.P., Carr, T.H., Rieser, J.J., Bodenheimer, B.: Updating orientation in large virtual environments using scaled translational gain. In: Proceedings of the 3rd Symposium on Applied Perception in Graphics and Visualization. p. 21–28. APGV '06, Association for Computing Machinery, New York, NY, USA (2006). https://doi.org/10.1145/1140491.1140495, `https://doi.org/10.1145/1140491.1140495`

This figure "userCase_-inside_view.jpg" is available in "jpg" format from:

http://arxiv.org/ps/2105.10720v1

This figure "usercase_-_topView.jpg" is available in "jpg" format from:

http://arxiv.org/ps/2105.10720v1