The Redirected Walking Toolkit: A Unified Development Platform for Exploring Large Virtual Environments

Mahdi Azmandian* Timofey Grechkin* Mark Bolas*† Evan Suma*

*USC Institute for Creative Technologies †USC School of Cinematic Arts

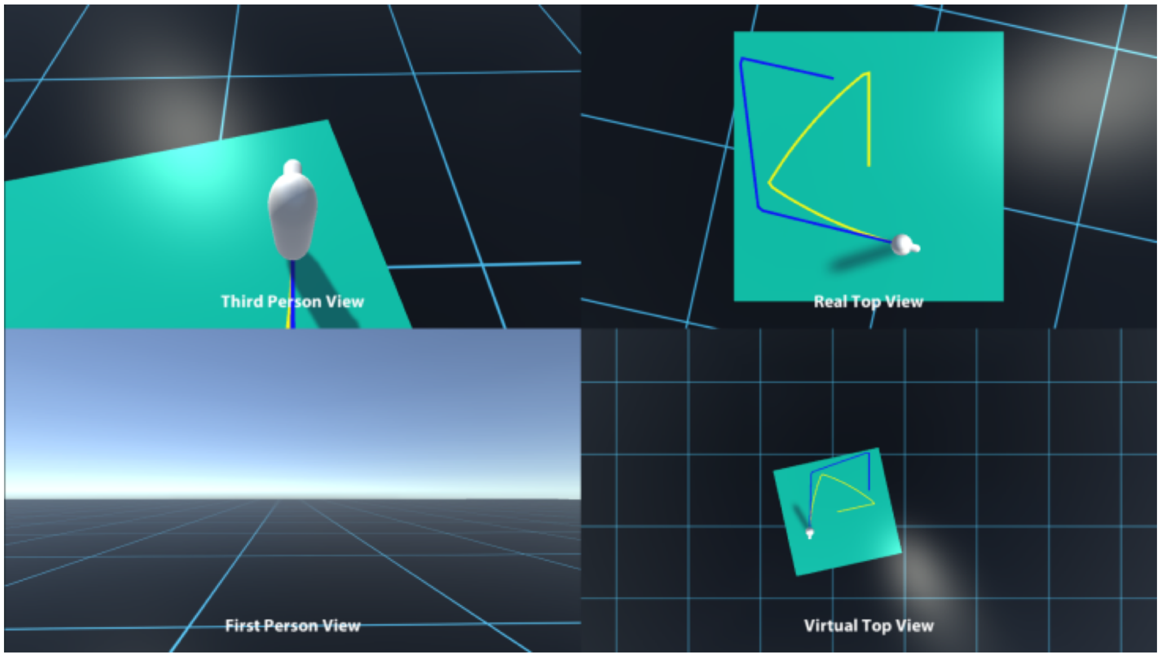


Figure 1: Snapshot of the Redirected Walking Toolkit simulating a user being redirected in a conceptual virtual scene; shown from various vantage points. The user's trajectory in the real world is shown in yellow, and virtual trajectory is shown in blue.

Abstract

With the imminent emergence of low-cost tracking solutions, everyday VR users will soon experience the enhanced immersion of natural walking. Even with consumer-grade room-scale tracking, exploring large virtual environments can be made possible using a software solution known as redirected walking. Wide adoption of this technique has been hindered by the complexity and subtleties involved in successfully deploying redirection. To address this matter, we introduce the Redirected Walking Toolkit, to serve as a unified platform for developing, benchmarking, and deploying redirected walking algorithms. Our design enables seamless integration with standard virtual reality configurations, requiring minimal setup effort for content developers. The toolkit's flexible architecture offers an interface that is not only easy to extend, but also complimented with a suite of simulation tools for testing and analysis. We envision the Redirected Walking Toolkit to be a common testbed for VR researchers as well as a publicly-available tool for large virtual exploration in virtual reality applications.

Index Terms: H.5.1 [Information Interfaces and Presentation]: Multimedia Information Systems—Artificial, augmented, and virtual realities; I.3.6 [Computer Graphics]: Methodology and Techniques—Interaction techniques; I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Virtual reality

*e-mail: {mazmandian, grechkin, bolas, suma}@ict.usc.edu

2016 IEEE 2nd Workshop on Everyday Virtual Reality (WEVR)
20 March, Greenville, South Carolina, USA
978-1-5090-0840-7/16/$31.00 ©2016 IEEE

1 Introduction

In the past few years, with the proliferation of head-mounted displays with motion tracking, VR is now evolving into a consumer-level commodity. The VR audience now runs the gamut from casual gamers, to museum exhibitors, to even elementary school students. Almost anyone with an idea, a laptop and few hundred dollars to spare, can join the VR development community and bring their vision to life. This has lead to an unprecedented quantity and variety of immersive experiences now being made available to the public on both desktop and mobile platforms. However, since consumer-grade HMDs typically come with limited (or even without) positional tracking, the prevalent image of seated-VR leaves something more to be desired.

Incorporating natural locomotion in VR has benefits beyond expanding the range of virtual reality applications. Research has shown that using a natural walking metaphor results in an enhanced sense of presence [17] and efficient navigation [11, 16]. Furthermore, users who experience walking in an environment, have improved spatial awareness and can develop better cognitive maps of virtual worlds [12].

Systems such as the Valve Lighthouse and Oculus Rift, provide low-cost room-scale tracking solutions that allow walking in virtual reality applications. The Valve Lighthouse as an example, can provide a 15 by 15 feet tracked space with simple installation procedures, making it practically a portable solution for VR, eliminating the need for extensive camera adjustment and calibration procedures that are common with traditional tracking solutions. This is a significant step towards enabling a greater variety of immersive experiences; though it also raises an important practical question: "Will exploring virtual worlds now be restricted to a single room?" And given the nature of many virtual reality experiences, must we regress to using joysticks again to explore large virtual environments, even when room-scale tracking is accessible?

Luckily, large budget allocation is not the only solution known to VR researchers. Redirected Walking [9] introduces subtle discrepancies between the user's motions in the real world, and what is perceived in the virtual world. This allows the user's trajectory in the real world to be different from the virtual trajectory without users noticing. The divergence of trajectories can be leveraged to explore a large virtual environment while in reality, being contained in a small tracked space. Therefore redirected walking can be viewed as a purely software-level solution that can expand the capabilities of small scale tracked space, enabling large virtual exploration at much less cost to the end user. In this work we provide tools to easily incorporate this software solution into common virtual reality applications.

2 Background on Redirected Walking

Redirected Walking was introduced over 15 years ago [9], and since its inception, a majority of the research has been on validating, evaluating limits and capabilities, and testing additional manipulation techniques under specific laboratory conditions. Various algorithms have been introduced, each employing different strategies for manipulating the user's trajectory to effectively keep them within the tracked area bounds. These algorithms can be categorized as *reactive* and *predictive*. Reactive algorithms are essentially greedy algorithms that make decisions based on the current state of the user at each point, and try to make the optimal choice based on a particular heuristic. For instance, the most commonly used algorithm in this category, Steer-To-Center, focuses on steering the user towards the center of the tracked space. This approach can be contrasted to a few recently introduced predictive algorithms [19, 7, 4] that predict the user's path in the virtual environment and use it to plan a redirection strategy.

Redirection algorithms can help with compressing a large virtual trajectory into a small space but they cannot guarantee the user safely remaining within the tracked space boundaries. Thus when the user inevitably reaches a physical boundary, a fail-safe mechanism must be triggered to prevent the user from leaving the tracked space. This safety measure is known as a *reorientation technique*, and was originally introduced and named *reset* by Williams et al [18]. The 2:1-Turn is the most commonly used form of reset, that instructs the user to perform a 360 rotation in place while scaling the virtual rotation by a factor of 2, resulting in a 180 degree rotation in the real world. Thus by the time the reset task is complete, the user will be facing in toward the tracked space.

Redirection algorithms and resets were first combined by Peck et al [8] as a *large-scale locomotion interface*. This system used Steer-To-Center with a reorientation technique that used *distractors*. Though this presented a complete workable pipeline for Redirected Walking, in practice, on average users encountered a reset every 5 meters of walking (in a 6.5$\times$6.5 meter tracked space) reporting it was bothersome and disruptive to the virtual narrative.

Studies have shown that predictive algorithms outperform reactive algorithms, specifically by reducing the frequency of resets. Therefore the redirection pipeline can be improved by replacing reactive algorithms with predictive ones. Though predictive algorithms have no evolved sufficiently to be readily deployable as with reactive methods, and require solving important subproblems such as automated virtual path prediction to be universally applicable. Furthermore, these methods are substantially more complicated to implement, and since crucial engineering subtleties are not addressed in the existing literature, faithfully replicating these methods is a great challenge for end users. And finally, no formal evaluation has been presented to compare existing predictive algorithms, to provide a clear guideline for which method to be used by developers.

In summary, Redirected Walking has been demonstrated as a promising low-cost solution for enabling large virtual exploration via natural locomotion, though it has not become a practical solution. We wish to address this matter, by bridging the gap between virtual reality researchers and developers. We present the Redirected Walking Toolkit as a unified platform for developing, benchmarking, and deploying redirected walking algorithms. The goal is to provide a publicly-available standard tool with flexible architecture that can easily be extended by both research scientists and the virtual reality community.

3 Toolkit Overview

The purpose of a Redirected Walking Toolkit is addressing the *practical* concerns and needs of this field. The most common of these cases are listed below:

- VR content creators needing a simple interface for deploying redirected walking that does not requiring knowledge of low-level implementation
- VR researchers wanting to extend this field by testing new solutions and benchmarking their work against previous methods
- VR end users inquiring about space requirements and expected performance for their specific configuration

With these use cases in mind, we outline a list of requirements for our Redirected Walking Toolkit:

- A standard implementation of existing redirection algorithms, both reactive and predictive
- A publicly available open-source code base available for expansion to the VR community
- A plug and play pipeline that can be deployed with minimal development effort that can guarantee user safety
- Providing a flexible architecture that allows for convenient extension
- A simulation platform with evaluation tools for both cost-benefit analysis (for end users) and also early testing of newly developed extensions and algorithms

Aiming to fulfill these requirements, we present version 0.3 of our toolkit as a package for the Unity3D[1] authoring environment, composed of a set of classes within a package along with a hierarchy of game objects to be used in Virtual Reality applications. The key features provided are redirection, reorientation, simulation, and analysis tools which we will expand on in detail in the next chapters. We will then continue by providing specific details of this version of the package, and also our deployment experience with the toolkit.

4 Redirection

4.1 Gains

Manipulating the relation between real and virtual motions is achieved by applying *gains*: injecting translations and or rotations as the user moves in a virtual environment. Three different types of gains have been identified in the literature [15]: (1) translation gains, (2) rotation gains, and (3) curvature gains. *Translation* gains involve scaling the user's translations, resulting in a perceived faster or slower displacement in the virtual world. *Rotation* gains apply scaling to the user's rotations, effectively increasing or decreasing perceived rotations. *Curvature* gains also involve inducing rotations, but instead are applied during translation. This is normally applied as a user walks forward towards a point in the virtual world, resulting in a curved path towards the target while perceiving a straight walking path. A redirection algorithm typically uses some combination of these gains to steer the user away from the boundaries of the tracked space.

4.2 Algorithms (Redirectors)

The current version of the toolkit provides enhanced versions of 2 classic reactive algorithms and 1 predictive algorithm that were all initially proposed by Razzaque et al [9]. Our reactive algorithms are Hodgson et al's [6] enhanced implementations of the widely used Steer-To-Center (S2C) and Steer-To-Orbit (S2O) algorithms. These techniques use rotation and curvature gains in a greedy approach to steer the user either towards the center (as in S2C) or in an orbit about the center (as in S2O) of the tracked space.

For our predictive algorithm, we have taken Razzaque's original zig-zag technique [10] that only relied on rotation gain, and expanded it to also support curvature and translation gain for maximal redirection. This method takes a zig-zag shaped virtual path and collapses it into a simple back-and-forth path between two points in the tracked space. We have also integrated a counter-deviation algorithm [2] that can gracefully manage moderate user deviation from the expected path, for a more robust and reliable solution.

4.3 Perceptual Thresholds

Redirection is meant to be unnoticeable, but users can become aware of manipulations if extreme gains are applied. Therefore our system by default ensures gain values are kept under empirically calculated noticeability thresholds. Steinicke et. al [13] showed that users can be turned physically about 49% more or 20% less than the perceived virtual rotation, distances can be downscaled by 14% and up-scaled by 26%, and users can be redirected on a circular arc with a radius greater than 22m while they believe they are walking straight.

In some cases, it is desirable to prioritize redirection efficacy over perceptibility, applying stronger gains that can better redirect users at the price of potentially compromising noticeability. To support this functionality, we provide options for manually tuning the thresholds for adjusting maximum and minimum permitted values for translation, rotation and curvature gains.

5 Reorientation

5.1 Safety Trigger

Redirection used in solitude is not sufficient to ensure users will remain within tracked space boundaries. When users are on the verge of leaving the tracked space they must be instructed to return. This is implemented by placing a safety trigger inside and within a distance (by default 0.5 meters) from each side of the tracked area boundary. When the user moves beyond a safety trigger, an instruction appears to guide the user to safety. The distance between the safety trigger and the boundary acts as a buffer to allow for users to react to notifications before reaching a potentially dangerous hard physical limit.

5.2 Resetters

Once a user passes the safety trigger, a reset must be activated to reorient the user back to the safe area. The current version of the toolkit supports the most widely used reset: the 2:1-Turn proposed by Williams [18]. The 2:1-Turn instructs the user to perform a 360 rotation in place while scaling the virtual rotation by a factor of 2, resulting in a 180 degree rotation in the real world. Thus by the time the reset task is complete, the user will be facing inward the tracked space. This method can also be seen as rotation gain applied at the boundary, mapping a 180 degree real rotation to a 360 virtual rotation. As a result, the user can resume walking in the intended direction prior to the reset, which now maps to the opposite direction in the tracked space.

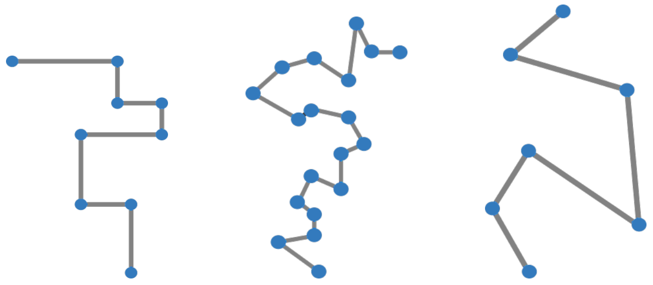


Figure 2: 3 categories of virtual paths. From left to right: Building Navigation, Small Exploration, and Large Exploration.

6 Simulation

Our toolkit supports simulating walking with redirection in addition to deployment with live users. Simulations can be used not only to aid with better understanding how various components of the system interact, but also to approximate realistic conditions for comparison and analysis purposes. The simulation component in version 0.3 of our toolkit is inspired by [3] and is comprised of two sub-components: A *Simulated Walker* mimicking a user walking in a tracked space, and also a *Virtual Path Generator*, that dictates the user's path in the virtual world.

6.1 Simulated Walker

A simulated walker is essentially an abstract avatar that represents a user navigating a virtual environment by moving in a tracked space. This movement can be either controlled via keyboard input or by the simulation itself, similar to an auto-pilot metaphor. The *keyboard input mode* supports basic actions such as walking forward, backward, and laterally, rotating (in-place) to the left and right, and also looking up and down. In *auto-pilot mode*, the walker navigates from one point of interest to the next based on a given sequence of waypoints. This is performed by moving forward at a fixed speed of 1 meter per second from one waypoint to the next. Once a waypoint is reached, the walker rotates in place at a constant rate of 90 degrees per second to face the next waypoint, at which point the forward walking is resumed; until eventually all waypoints are cleared. Also when a reset is triggered, an override mechanism is enabled to follow the specific instructions of the applied reset. For instance, in the case of a 2:1-Turn reset, the user stops and rotates in place until the reset task is complete.

6.2 Virtual Path Generator

A user's trajectory in a virtual environment is implicitly dictated by the virtual environment's layout. Therefore to simulate walking in various virtual environments, instead of creating a suite of environment layouts, we generate various virtual trajectories that would correspond to walking in various virtual environments. This simplifies our design by eliminating the intermediary step of inferring the navigability of a virtual environment in order to deduce possible virtual trajectories.

Our system can procedurally generate a variety of random paths that aim to replicate realistic walking trajectories in a virtual environment. Each virtual path is defined as a series of waypoints. In addition to the overall length of a virtual path, the distance between consecutive waypoints and the angles formed by connecting them can be controlled by adjusting the parameters of our virtual path generator. By default our system provides support for generating 3 types of random paths: 1) Small Exploration 2) Large Exploration 3) Building Navigation (see Figure 2).

6.3 Simulating Time & Framerate

When our system is in simulation mode, one of the important resources we can save on is time. Our system provides *time simulation* feature that allows for faster calculation of a simulation's outcome. Instead of using actual time elapsed between calculation frames, we can programmatically set this value, which enables simulating arbitrary framerates. For instance, to simulate a 60Hz framerate, we artificially set the time elapsed between frames to $\frac{1}{60}$ seconds. And since the real time elapsed is typically smaller than this value, the execution time is reduced. Furthermore the simulation can be run in *test mode* that omits the rendering pipeline to speed up the calculation even further (in our experience by up to an overall factor of 100). Therefore with simulated time we can run simulations at much higher speeds, and also controlled framerate conditions.

7 Analysis Tools

7.1 Path Visualizations

The most informative visualization tool for redirected walking is comparing real and virtual trajectories. Our toolkit visualizes real and virtual trajectories from various vantage points such as first and third person, in addition to bird's eye view from both a fixed real and virtual world (see Figure 1). Fixing the real and virtual references provides a better perspective of the path in the real world, specifically delineating how the real and virtual worlds move with respect to each other. High resolution snapshots of these visualizations are generated at the end of each virtual path and can also be created on demand at runtime.

7.2 Metrics and Logging

For more comprehensive analysis, our system has built-in features for keeping track of various behavioral statistics and generating log files. The metrics currently supported in our system are:

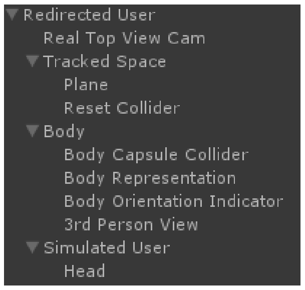
- total reset count
- real and virtual distance travelled between resets
- time elapsed between resets
- overall gains applied
- overall real and virtual distance travelled
- real and virtual position at custom intervals
- applied gain at custom intervals

Though the most common performance metrics are a function of resets, researchers often also consider minimizing overall gains applied, even if these gains are unnoticeable. Furthermore logging position and applied gain information can be useful for replicating specific moments in a simulation for deeper diagnosis and assessment.

7.3 Batch Testing

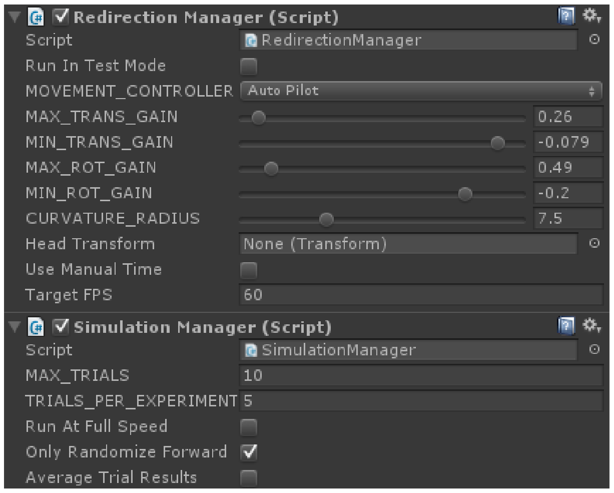
To simplify executing multiple simulations across various conditions, our system supports batch testing. In this mode, the user can define various conditions that will be executed in succession. In version 0.3 of the toolkit, the factors that define a condition are as follows:

- RDW algorithm
- reset type
- virtual path category
- tracked space dimensions
- simulated walker type



```
▼Redirected User
   Real Top View Cam
  ▼Tracked Space
      Plane
      Reset Collider
  ▼Body
      Body Capsule Collider
      Body Representation
      Body Orientation Indicator
      3rd Person View
  ▼Simulated User
      Head
```

Figure 3: Arrangement of toolkit game objects in scene graph.



```
▼ ☑ Redirection Manager (Script)
  Script                   RedirectionManager
  Run In Test Mode         ☐
  MOVEMENT_CONTROLLER      Auto Pilot
  MAX_TRANS_GAIN           0.26
  MIN_TRANS_GAIN           -0.079
  MAX_ROT_GAIN             0.49
  MIN_ROT_GAIN             -0.2
  CURVATURE_RADIUS         7.5
  Head Transform           None (Transform)
  Use Manual Time          ☐
  Target FPS               60
▼ ☑ Simulation Manager (Script)
  Script                   SimulationManager
  MAX_TRIALS               10
  TRIALS_PER_EXPERIMENT    5
  Run At Full Speed        ☐
  Only Randomize Forward   ✓
  Average Trial Results    ☐
```

Figure 4: Example of customizable redirection and simulation options by simple adjustment parameters in the Unity3D editor.

- perceptual thresholds

Each condition can also be repeated an arbitrary number of trials. Note that currently the only element of randomness present in the system is derived by virtual path generator, therefore each trial of the same condition, will generate a different virtual path from the same category.

8 Software Package Details

8.1 Main Components

Our toolkit is encapsulated in a Unity3D package.The sample scene included is set with default configurations that can easily be customized with the exposed parameters in the editor (see Figure 4). The root game object that contains the representation of the user and the tracking space is named "Redirected User". When redirection is applied, this game object changes position and orientation, and by inheritance, so does the user and tracking area. Redirected User is in essence, the origin of the real world with respect to the virtual world (see Figure 3). Therefore the application of redirection is essentially moving the reference of the real world in relation to that of the virtual. Redirection is applied by a *Redirector* script and resetting is applied by a *Resetter*, whic are both attached to Redirected User. These scripts are managed by the RedirectionManager which serves as a hub, that connects scripts with the resources necessary such as the position of the user, and also notifies them of changes such as the user reaching a boundary.

The script that controls simulations is *SimulationManager*, which enables user movement via keyboard input (via the *KeyboardController* script) or by the toolkit itself (via *Simulated-*

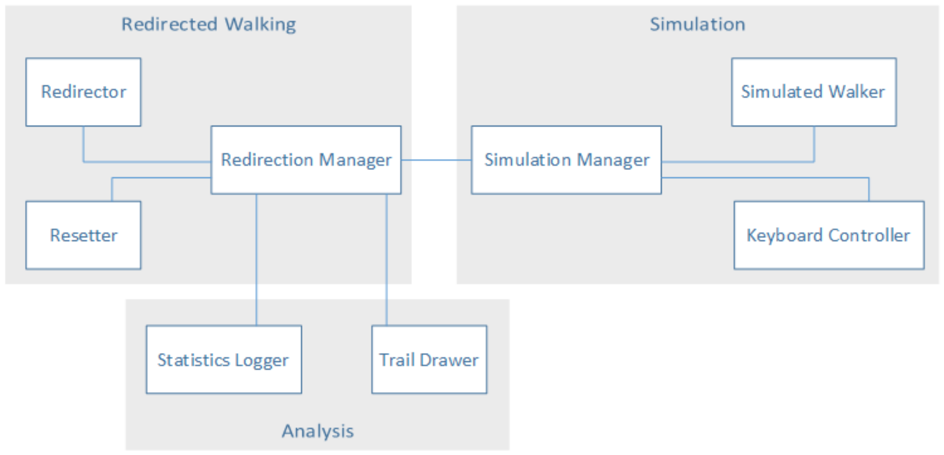


Figure 5: Components of the Redirected Walking Toolkit and how they are connected.

Walker). SimulationManager also provides functions for running tests under various conditions, and generating various virtual paths for the simulated user to follow.

Analysis tools are provided by the *TrailDrawer* and *StatisticsLogger* scripts. TrailDrawer creates a trail of the user's path with respect to the tracked space (real world) and the virtual world. A snapshot of the user's trajectory is taken from various vantage points and logged to file at the end of the each simulation, and can also be generated at any point of execution with a single button press.

8.2 Extension

The redirected walking toolkit is with layers of abstraction and a modular paradigm which implicitly provide guidelines for easy extension. As an example, a new redirector can be introduced by extending the *Redirector* abstract class, and implementing the required abstract functions with the use of available helper functions. In this instance, a redirector is required to implement the *ApplyRedirection* function which is called at the proper time in the execution pipeline, and can easily be implemented by using simple calls such as *ApplyRotationGain*. Currently Redirector is extended by *SteerToRedirector* which is in turn extended by both *S2CRedirector* and *S2ORedirector*. A similar design paradigm holds for extending the *Resetter* abstract class which is extended by *TwoOneTurnResetter*.

8.3 Deployment

Incorporating redirection into an existing project can be accomplished in three simple steps: 1) Drag the toolkit prefab into the game scene 2) Drag the user's head transform to its reference in the RedirectionManager script 3) Set the *Tracking Area* object's scale to match the available tracked space. Installation instructions, tutorials, and the source code can be found at `http://projects.ict.usc.edu/mxr/rdwt/`.

9 Deployment Experience

[t] Our toolkit makes no assumptions about the target platform's hardware configuration, which is the key to its versatility. A testament to this has been our success in integrating the toolkit with 3 different HMDs (Oculus, Vive and Wide5) and 4 tracking systems (Valve Lighthouse, Vicon, OptiTrack and PhaseSpace).

To date the redirected walking toolkit has been publicly used to create two live-user experiences (Redirected Viking [14], and Near-Field VR [5]) and also provide new insight on physical space requirements of redirected walking [3].

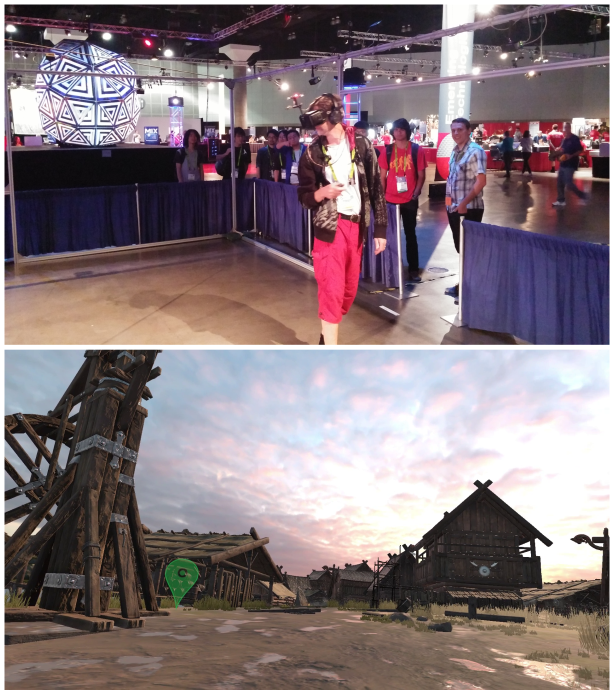


Figure 6: The Redirected Viking Demo. User exploring large Viking Village environment (bottom) in a physical tracked space of 7×7 meters (top).

9.1 Redirected Viking

To showcase the simple drag-and-drop deployment pipeline our toolkit offers, we demonstrated integrating redirected walking with the "Viking Village", a standard virtual environment provided by Unity3D. By following the simple 3 step procedure, we enabled free exploration of this generic virtual environment, redirecting users (with Steer-To-Center) keeping them within the tracked space boundary, and triggering resets when necessary. This demonstration was also expanded to showcase new resetting mechanisms we introduced that blend with the virtual narrative for less disruptive reset prompts. Over the course of 3 days, more than 70 people experienced this demo at SIGGRAPH Emerging Technologies [14] (see Figure 6).

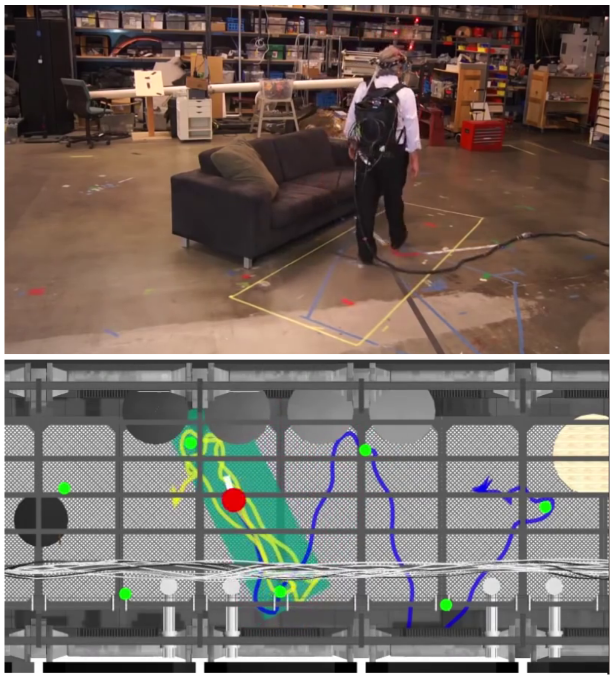


Figure 7: The Near-Field Demo. User visiting points of interest in a 12×4 meter environment (bottom) by walking back and forth in a 3.5×1.2 tracked space(top).

9.2 Near-Field VR

Though reactive algorithms are suitable for free exploration, taking a predictive approach can substantially reduce space requirements. We leveraged this key feature when we were tasked to fit a 12×4 meter virtual environment in a 3.5×1.2 meter tracked space for a Near-Field VR experience. This was achieved by reconfiguring the arrangement of points of interest in the experience to make it applicable to our zig-zag algorithm. We debut this work on a small stage before hundreds of computer graphics enthusiasts, winning first prize in the SIGGRAPH Immersive Realities contest [5] (see Figure 7).

9.3 How Shape and Size Affect Performance

One of the most practical concerns for end users the amount of space required for successfully using redirection. With our toolkit, we managed to systematically evaluate how the shape and size of the tracked area affects performance and provide new insight on the optimal tradeoff between cost and performance [3]. This was made possible by using simulations that enable testing redirection under various condition with lengthy trials and numerous repetitions, to control for many interacting factors in a redirected walking setup.

10 Conclusion and Future Work

Redirected Walking is a powerful low-cost solution for enabling natural locomotion in large virtual environment explorations. The wide adoption of redirection is hindered by the barrier to entry due to implementation difficulty and deployment complexity of redirected walking. In this work we have presented the Redirected Walking Toolkit to address this matter, and facilitate collaboration between researchers and also end developers.

Though our work has already been deployed in publicly demonstrated experiences, there are many features we plan to add in upcoming versions. First, we wish to incorporate a more realistic representation of a walking user by introducing elements such as noisy movement, gait oscillations and random gaze aversion. Such changes can substantially improve the validity of simulated user experiments, and also test the robustness of predictive algorithms to unaccounted user deviations. Also we wish to introduce tools that can infer the implied navigation network of a given virtual environment to create virtual paths that better represent real human trajectories and capture nuances of a virtual environment's architecture. Finally, we plan to provide a broader range of algorithms for future deployment and analysis.

The Redirected Walking Toolkit is a unified platform for developing, benchmarking, and deploying redirected walking algorithms. Our source code has been made publicly-available to be easily extended by research scientists and the virtual reality community. We hope this work opens the door to deeper collaboration in this field and ultimately lead to redirected walking becoming a standard feature of motion tracking systems.

References

[1] Unity3D Game Engine. `https://unity3d.com/`. Accessed: 2016-02-02.

[2] M. Azmandian, M. Bolas, and E. Suma. Countering user deviation during redirected walking. *In Proc. the ACM SAP*, page 4503, 2014.

[3] M. Azmandian, T. Grechkin, M. Bolas, and E. Suma. Physical Space Requirements for Redirected Walking: How Size and Shape Affect Performance. *In Proc. ICAT-EGVE*, 2015.

[4] M. Azmandian, R. Yahata, M. Bolas, and E. Suma. An Enhanced Steering Algorithm for Redirected Walking in Virtual Environments. *In Proc. IEEE VR*, pages 65–66, 2014.

[5] M. Bolas. In *ACM SIGGRAPH 2015 Computer Animation Festival*, page 193, New York, NY, USA. ACM.

[6] E. Hodgson and E. Bachmann. Comparing four approaches to generalized redirected walking: simulation and live user data. *IEEE TVCG*, 19(4):634–43, 2013.

[7] T. Nescher, Y.-Y. Huang, and A. Kunz. Planning Redirection Techniques for Optimal Free Walking Experience Using Model Predictive Control. *3DUI 2014*, pages 111–118, 2014.

[8] T. C. Peck, H. Fuchs, and M. C. Whitton. The design and evaluation of a large-scale real-walking locomotion interface. *IEEE TVCG*, 18(7):1053–1067, 2012.

[9] S. Razzaque. *Redirected Walking*. PhD thesis, Chapel Hill, NC, USA, 2005.

[10] S. Razzaque, Z. Kohn, and M. C. Whitton. Redirected Walking. *In Proc. EUROGRAPHICS*, pages 289–294, 2001.

[11] R. Ruddle and S. Lessels. The benefits of using a walking interface to navigate virtual environments. 16(1):1–18, 2009.

[12] R. Ruddle, E. Volkova, and H. Bulthoff. Walking improves your cognitive map in environments that are large-scale and large in extent. 18(2):1–20, 2011.

[13] F. Steinicke, G. Bruder, J. Jerald, H. Frenz, and M. Lappe. Estimation of Detection Thresholds for Redirected Walking Thechniques. *IEEE TVCG*, 16(1):17–27, 2010.

[14] E. A. Suma, M. Azmandian, T. Grechkin, T. Phan, and M. Bolas. In *ACM SIGGRAPH 2015 Emerging Technologies*, pages 16:1—-16:1, New York, NY, USA. ACM.

[15] E. A. Suma, G. Bruder, F. Steinicke, D. M. Krum, and M. Bolas. A taxonomy for deploying redirection techniques in immersive virtual environments. *In Proc. IEEE VR*, pages 43–46, 2012.

[16] E. A. Suma, S. Clark, S. L. Finkelstein, and Z. Wartell. Leveraging Change Blindness for Walking in Virtual Environments. *In Proc. IEEE VR Workshop on Perceptual Illusions*, page 10, 2010.

[17] M. Usoh, K. Arthur, M. C. Whitton, R. Bastos, A. Steed, M. Slater, and F. P. Brooks. Walking > walking-in-place > flying, in virtual environments. *In Proc. SIGGRAPH 1999*, pages 359–364, 1999.

[18] B. Williams, G. Narasimham, B. Rump, T. P. McNamara, T. H. Carr, J. Rieser, and B. Bodenheimer. Exploring large virtual environments with an HMD when physical space is limited. *In Proc. APGV 2007*, 1(212):41, 2007.

[19] M. A. Zmuda, J. L. Wonser, E. R. Bachmann, and E. Hodgson. Optimizing constrained-environment redirected walking instructions using search techniques. *In Proc. IEEE TVCG*, 19(11):1872–1884, 2013.