Procedural Terrain Generation for Medical Rehabilitation

Michael Andereck
The Ohio State University
CSE Department
395 Dreese Laboratories
2015 Neil Avenue
Columbus, Ohio 43210-1227
andereck.3@osu.edu

Alan Price
The Ohio State University
Advanced Computing Center
for the Arts and Design
1224 Kinnear Rd
Columbus, Ohio 43212
aprice@accad.osu.edu

Roger Crawfis
The Ohio State University
CSE Department
395 Dreese Laboratories
2015 Neil Avenue
Columbus, Ohio 43210-1227
crawfis.3@osu.edu

ABSTRACT

Virtual terrain generation has been a popular area of research over the last thirty years. More recently the topic of using video games to promote exercise and rehabilitation has gained momentum. We propose a system which allows for physical therapists to aid in the creation and management of virtual environments for use in conjunction with walking and balance exercises. Our system allows the therapist to design a 3D path with challenges including hills and turns suitable for the ability of the patient. After the path has been designed, the system generates a terrain using an iterative surface-matching quadtree algorithm, culminating in an immersive, engaging environment for the patient.

Keywords

procedural terrain generation; user-defined paths; immersive environments; video games

1. INTRODUCTION

Virtual environments are increasingly used in exercise-based video games and rehabilitation systems. As users perform exercises, they are able to receive real-time feedback and are entertained, leading to more enjoyment and engagement during the activity. In our rehabilitation scenario, the patient walks on a treadmill with the environment projected in front of them, mimicking the speed and orientation of the walker. This virtual environment is key to keeping the patient engaged in the rehab process. Another key is the ability of the rehabilitation operator to tune the exercise regimen to the needs of the patient. If the patient is just beginning to regain locomotion, they would not need to walk up a steep hill. At the other extreme, a patient who is finishing their rehabilitation would likely need a more strenuous challenge than walking through a flat field. Needs vary from patient to patient and even from week to week. We cannot forsee all necessary path configurations ahead of time, so a customizable solution is required.

One way to customize paths and environments is to hand model everything. While this approach would work in theory, in reality the cost of generating all the necessary environments by hand would be prohibitive. For example, a sample terrain designed by hand took over 40 hours to model and texture. In this paper we present a system which can be used by a physical therapist with little knowledge of computer graphics to design a path and virtual environment of comparable scale and complexity in less than an hour.

Our system begins with a path definition. This path is used to determine the exercise for the patient. The therapist (operator) decides how long the path is as well as what shape it has, with the ability to design inclines, declines, turns, and cycles. These features are linked to a treadmill on a motion platform which can rotate and tilt. As the path inclines, so does the treadmill. This provides the patient with a realistic physical challenge to match their visual experience.

Once the path is defined, an environment is procedurally generated around the path. Rather than the real-world approach which encounters a hill and places a road to conform to the hill, our system sees an inclined path and creates a virtual hill to match it. While this would be a fairly trivial task for a path which is linear, our paths can turn and loop around, leaving interesting landscape features in their wakes.

The rest of the paper is organized as follows. In Section 2 we look at work related to the idea of procedural terrain generation and physical therapy. Section 3 introduces and describes the process of designing paths. Section 4 describes the algorithm for conforming a landscape into the shape prescribed by a path. The last sections show results, conclusions, and future work.

2. RELATED WORK

Many techniques exist for synthesizing terrains. Procedural terrain modelling has been an increasingly popular topic of research for the last thirty years. The most common techniques for synthesis include geometric modeling and physical simulation. Geometric models are commonly based on fractal designs such as those used by Mandelbrot [13], Fournier [10], and Perlin [15]. Physical simulation techniques are largely based on erosion methods such as thermal erosion and fluvial erosion. Both are discussed by Musgrave et al [14]. These techniques can produce realistic looking terrains, but are often difficult to control.

One method for recreating realistic terrains is proposed by Zhou et al [21] which applies real-world elevation data to user-defined control paths. Thus a mountain range or canyon can be conformed to follow any desired route. Several other works allow users to sketch splines, rivers, mountains and other features and then generate terrains to match [1, 11]. These techniques are designed for the user to define terrain features such as ridges which are then directly built in to the terrain shape. Our paths are not necessarily features of the terrain and will not always be ridges or valleys.

A technique for constrained fractal terrains has been proposed by Belhadj [2]. This idea was created to reconstruct digital elevation maps by supersampling compressed elevation sets gathered by satellites or other means. They are also able to account for user sketches as constraints for terrain generation. They use a triangle-based Midpoint Displacement Bottom-Up process (MDBU) to fill in the gaps between known control points and those which require additional constraint. This technique does a fine job in filling in the gaps between user-defined control points, but has complicated parameter settings which would cause a steep learning curve for a novice user. In an earlier work [3], they allowed users to define mountain ridges and river valleys to give an initial shape to the terrain. This technique produces terrains by refining fractals between the constraints, but is not suitable for our needs as the terrain always slopes down from the ridges and up from the valleys. We want to use the shape of the path to inform the terrain builder of the slope of surrounding features without having the path designer specify whether they are on a ridge or in a valley.

In our research we have found limited work which adapts the shape of a terrain to a pre-defined path. One author proposed the idea but in their actual work did not end up using paths [5]. Another work by Smelik et al [18] uses an integration of manual editing and procedural generation to create landscapes at low cost to the user but with more control than purely procedural methods. They also point out that procedural content is well-suited for large-scale operations while manual editing works well with small details, but it is difficult to bridge the gap with medium levels of detail and control. Their design is meant to be edited continuously with live updating of features such as cities, rivers, and roads. Because of the interaction between these components, strict manually-designed features are not well-preserved upon procedural generation.

Another paper by Stachniak and Stuerzlinger [19] presents a method for conforming an existing terrain to the shape of a constraint which could be defined as a path or road. This method begins with a defined terrain and a path for constraint, and then seeks to take steps toward an optimal solution by deforming segments of the terrain under a gaussian kernel. While this technique can come close to an optimal solution, there is no guarantee of convergence. This technique involves manipulating an existing terrain, whereas we are working from scratch. One more technique for applying local manipulation to an existing terrain is presented by Bruneton et al [7]. They begin with large-scale Geographical Information Systems (GIS) data and render is using a level of detail (LOD) approach, focusing on local manipulation of the terrain from user-defined roads and rivers. Once again this system is designed for use with a pre-existing terrain.

Taking the opposite approach, there has been work done related to finding a suitable path using the terrain as the constraint [12]. Given a terrain with origin and destination points, Galin et al are able to generate a road connecting the two while considering obstacles including rivers and hills. Our work seeks to do the reverse: we are designing a path and then generating a terrain shape to conform to that path. In our case, the path is the key component because of its link to the therapy, while the environment is secondary.

Related to our motivation for immersing the patient in a virtual environment, gameplay has been shown to motivate physical activity [20]. Significant work has been done in research on exergames, the design of video games used in conjunction with exercise equipment. Considerations for these games are discussed in [17]. Rewards for physical interaction has been shown to increase phyical activity while playing games without decreasing the enjoyment of the experience [4]. In a smiliar vein, Bianchi et al [6] demonstrated that body movement and physical activity not only increase the level of engagement for players by motivating a sense of presence in the digital world, but the experience became more rewarding for players as a result.

While it has been shown that games can serve to assist in promoting physical activity, they cannot provide all the motivation on their own. Chandra et al [8] showed that while patients enjoyed their individual exercise sessions more while using motivation through entertainment, they still wanted to see long-term progress numbers toward thier overall rehabilitation.

Dimovska et al [9] demonstrate that adapting level difficulty based on previous performance can positively impact rehabilitation goals. Using a Wii Balance Board, they developed an adaptive skiing game which placed future gates based on the player's ability as shown through prior gate passing. By adapting the difficulty to the patient ability, they are able to maintain better engagement of subjects because they are less prone to boredom in the case of easy levels, or frustration in the case of levels which are too difficult. Our system allows the physical therapist to create a path with an appropriate level of challenge for each patient.

3. PATH GENERATION

In order to have a rehabilitation tuned to the specific needs of a patient, we need to provide the operator control over the shape of the path the patient has to traverse. The path is defined by a set of control points which are user-placed. To interpolate between control points we use Hermite splines for a second-order continuity. Once the spline curve has been defined, we determine the shape of path geometry, and in Section 4 we use the path geometry to calculate the shape of the terrain.

3.1 Control Points

The main control the operator has in shaping the path is in the position of the control points. These points are shown as white spheres as in Figure 1. The operator has freedom to select a sphere and move it anywhere in the 3-dimensional world space of the game. Points may be added to the end

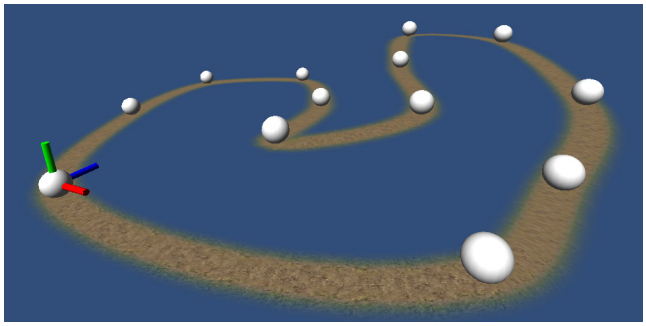

Figure 1: Sample of a small path using spheres as control points.

of a path, inserted into the middle of the path, moved, and deleted. As points are manipulated, the path is updated for real-time feedback. One element which can aid therapists in tuning the rigor of the workout is the display of path statistics such as the length of the path and the current slope. These numbers can be seen when editing control points or when walking along the path as seen on the right side of Figure 8. Additionally the operator can choose an open path which can start and end at arbitrary locations, or a closed loop path. There is no practical limit to the number of control points used in the path. The main limiting factor is the time required by the operator to place the path points.

3.2 Hermite Splines

With our control points in place, we need to interpolate between them to give shape to the path. Linear and quadratic interpolation do not provide the smoothness we require, so we use a cubic Hermite spline. We need the interpolation to be smooth because of the nature of the immersive environment. With a large, immersive display, jerky motion can cause disorientation of the user.

Our Hermite spline interpolates the path shape between each consecutive pair of path control points. Segment k is defined between control points p_k and p_{k+1}. To obtain the starting tangent for p_k, we use half the vector formed between the surrounding control points: $m_k = (p_{k+1} - p_{k-1})/2$ and similar for p_{k+1} our tangent is $m_{k+1} = (p_{k+2} - p_k)/2$. The automation of tangent vectors makes the system easier than requiring explicit tangents as in some drawing programs. Then for a given interval between control points, with interpolant $t\epsilon[0,1]$, we find the position $p(t)$ using the following equation:

$$p(t) = a * p_k + b * m_k + c * p_{k+1} + d * m_{k+1} \qquad (1)$$

where $a = (2t^3 - 3t^2 + 1)$, $b = (t^3 - 2t^2 + t)$, $c = (-2t^3 + 3t^2)$, and $d = (t^3 - t^2)$.

3.3 Character Position

When traversing the path through the world, the position of the viewer is constrained to the defined path. The position is determined by the parameters of the Hermite spline as in Equation 1, while the orientation of the character is based on an approximate tangent of that curve. This tangent is given as the normalized vector $\vec{p}_{t+\epsilon} - \vec{p}_t$. Sampling the path slightly ahead of the current position (at $t + \epsilon$) allows us to

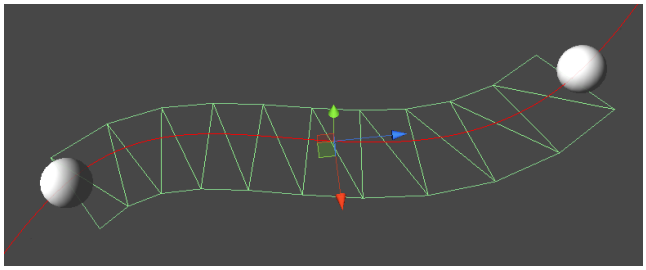

Figure 2: The individual path segments (green) follow the curve of the Hermite spline (red).

give the user a look-ahead anticipation of upcoming curves. For this reason the path needs to be second-order continuous. The first order ensures that the character's position does not jump from time to time, while the second order continuity means that the character's view direction does not take an abrupt turn.

3.4 Distance Interpolation

In order to create a realistic experience for the user, they must be able to move along the path at a consistent speed, regardless of the underlying control system. With our parameterized splines, the primary definition of position along the spline is consistent with the relative position between control points. In this case it takes a constant change in the parameter value t to interpolate from control point A to control point B.

Under the default parameters, it would take one second to move from point A to point B, and the same time to move from point B to point C. This time would not change regardless of the distance between each set of points.

Consider d as the distance along the curve, and say $\vec{p}$ is the position in space. We would like the following to hold:

$$\frac{d\vec{p}}{dd} = \delta \qquad (2)$$

that is, the change in spatial position relative to the change in the path distance parameter should be a constant (δ). We can sample the curve through time to construct a mapping from time to spatial distance. Now consider a function $\Phi : \mathbb{R} \Rightarrow \mathbb{R}$ such that $\Phi(d) = t$. We use the map to determine the translation for Φ. The Hermite interpolation function we will call $H : \mathbb{R} \Rightarrow \mathbb{R}^3$ which translates $H(t) = \vec{p}$. Combining those layers yields:

$$\vec{p} = H(t) = H(\Phi(d)) \qquad (3)$$

which gives us the ability to increment d at a constant rate. We are able to then apply the speed of the user on the treadmill to consistently determine the rate of the user through the world, satisfying Equation 2.

3.5 Path Geometry

The geometry of the path is extruded along the path spline, using normals from the spline to determine the vertices of the geometry. Figure 2 illustrates how the shape of the Hermite spline determines the shape of the corresponding segment of path geometry. Note that by default there is no roll to the path, that is the path is flat horizontally.

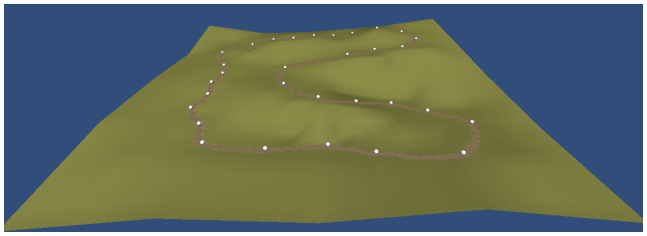

Figure 3: The terrain has been modified to match the desired path.

The resolution of the path segments is static from one control point to the next. Unlike the character position, the path geometry does not need to be normalized based on world space distances. This is because the importance of the path segments is in the overall shape of the segment, not in the individual details.

4. TERRAIN GENERATION

The main contribution of our work is the concept of moving from the shape of a user-defined path to fitting an entire terrain around that path. Our terrain is defined as a 2-dimensional height map. This map is scaled in the X and Z directions to account for the potential range of scales on the user-defined path. We define the Y-axis as the vertical scale along which the height values vary. We begin by matching the terrain to the heights of the path segments. Next we start an iterative plane-fitting process to bring the terrain into alignment with the path.

4.1 Matching the Path

To begin the terrain defining process, we match the terrain to the user-defined height of the path. One difficulty arises due to the difference in definitions for the terrain and the path segments. The terrain is defined as a grid-based height map, while the path is a ribbon of 3D geometry. Our target is the height map.

For each point in the height map we check for the presence of path geometry at the same XZ coordinate. If there is a collision with that geometry, we assign the corresponding height (Y) into the height map. We also mark that position as a part of the path so that it is preserved as a terrain-path point when we perform the plane fitting in the next section.

At this point we also modify the texture of the terrain to indicate where the path is. When walking through the world, it is helpful to have a visible path to follow, not just an invisible line to which you are anchored. To easily avoid z-fighting issues or other problems caused by imperfect matching between the path height and the terrain height, we simply apply the path texture to the terrain on the appropriate terrain-path positions.

4.2 Shaping The Terrain

Once we have rasterized the path, we can begin the process of matching the surrounding terrain to the shape modeled by the terrain-path. We have a few criteria for the resulting shape of the terrain:

1. Terrain adjacent to the path should match the height of the path.
2. Terrain should naturally interpolate between path segments.
3. Terrain should have shape outside the confine of the path. I.e. extrapolating the slopes.

We need to use a method to fit the terrain to the path at multiple scales. Several prior techniques rely on local manipulation of existing terrain shapes [18, 19]. We do not want to constrain our path generation process to be driven by an existing terrain shape, but to be open for whatever the physical therapist deems appropriate for the patient. Our technique is to iteratively find a best-fit surface. By finding the best fit at various scales via a quadtree [16], we are able to provide slopes over the scale of the entire terrain as well as refine local details to conform to the height adjacent to the path. We describe the technique of matching a single quad in Section 4.2.1 and then how to combine different levels of the tree in Section 4.2.2.

4.2.1 Linear Least Squares

In general we want to solve the overdetermined system defined by a set of m terrain-path points with n unknown coefficients.

$$\sum_{j=1}^{n} X_{ij}\beta_j = y_i, (i = 1, 2, ..., m). \tag{4}$$

Note that in the case of a planar equation, we will use 3 coefficients, giving us $n = 3$, but in general we may have more coefficients, β_1, β_2, . . ., β_n. In order to solve this system, we need $m \geq n$. As we use a different path or change levels of the quadtree, m will change. We can write Equation 4 in matrix form,

$$\mathbf{X}\boldsymbol{\beta} = \mathbf{y} \tag{5}$$

where

$$\mathbf{X} = \begin{bmatrix} X_{11} & X_{12} & \cdots & X_{1n} \\ X_{21} & X_{22} & \cdots & X_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ X_{m1} & X_{m2} & \cdots & X_{mn} \end{bmatrix}, \boldsymbol{\beta} = \begin{bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_n \end{bmatrix}, \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix} \tag{6}$$

Our objective is then to minimize the squared error term,

$$E(\boldsymbol{\beta}) = \sum_{i=1}^{m} |y_i - \sum_{j=1}^{n} X_{ij}\beta_j|^2 = \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 \tag{7}$$

Assuming the n columns of matrix $\mathbf{x}$ are linearly independent, we can find a unique solution to the minimization problem, namely,

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}^\mathbf{T}\mathbf{X})^{-1}\mathbf{X}^\mathbf{T}\mathbf{y} \tag{8}$$

This technique can be used to solve for the best fitting surface of any polynomial or parabolic shape. We have implemented the system to solve for the best fit plane. With a planar system, Equation 4 becomes:

$$y_i = \beta_1 x_i + \beta_2 z_i + \beta_3 \tag{9}$$

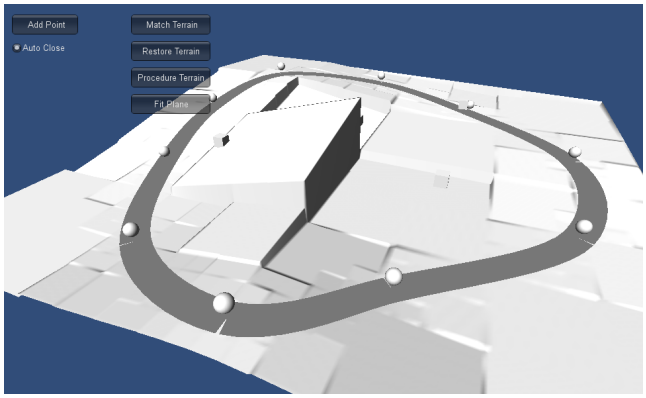

Figure 4: When the corners don't match up, the terrain loses its realism.

When we solve Equation 8 we have coefficients β_1, β_2, and β_3 to determine values of y in Equation 4. This defines the slope of the best fit plane required for one square of the terrain as shown in Figure 7B. Clearly one plane cannot fit the entire path unless that path is entirely flat. To accomodate this, we need to iterate down to multiple levels of detail using a quadtree approach, refining the shape of the terrain at multiple resolutions. Figure 7 demonstrates the refinement of a height map being fitted to a path. At each level, planes are fit to the terrain-path points they contain. Part B fits one plane over the full path. Part C fits four planes (one per quadrant), part D 16 planes, and so on. At each level, the area covered by each plane is reduced, creating a better fit height map.

4.2.2 Diamond-Square

We want to refine the fitting of the terrain, starting at the largest scale and moving to the smallest. Blending from one level of quadtree planar matching to the next is tricky. If we do not match the edges of the plane squares, we obtain some clearly incorrect results as in Figure 4. We therefore need to maintain some information of the points used from one level to the next.

The first level or plane fitting is always done over the entire terrain. This establishes the basic slope of the whole area. If the defined path has an overall inclination, we want to be able to maintain the feel that the terrain is on a hillside, rather than having it flatten on the edges beyond the scope of the defined path. This step is shown for a sample path in Figure 7B. Next we divide the terrain into quadrants and find the best fit planes for each of these based on the terrain-path points contained in each as described in Section 4.2.1.

Beyond the first iteration of plane matching, there will be a few issues to consider. One is that adjacent planes share boundaries. If we do not match those boundaries, we obtain a result as in Figure 4. On each level of iteration, we only compute the heights on the corners of each square as indicated in red in Figure 5. The heights of the corners applied to the height map are determined by averaging heights contributed by adjacent squares. Because of the contributions from multiple squares to a single corner, each square is not guaranteed to be planar. If the corner height shared by adjacent squares differ greatly, they will still be resolved on further iterations. This in particular arises when the path shape is not close to planar, such as the switchbacks found on the side of a steep slope. We may find multiple levels of a switchback path to be in the same square, meaning that a single plane cannot be a good representative for that section. This issue is resolved on further refinement as seen in Figure 7F as compared to C.

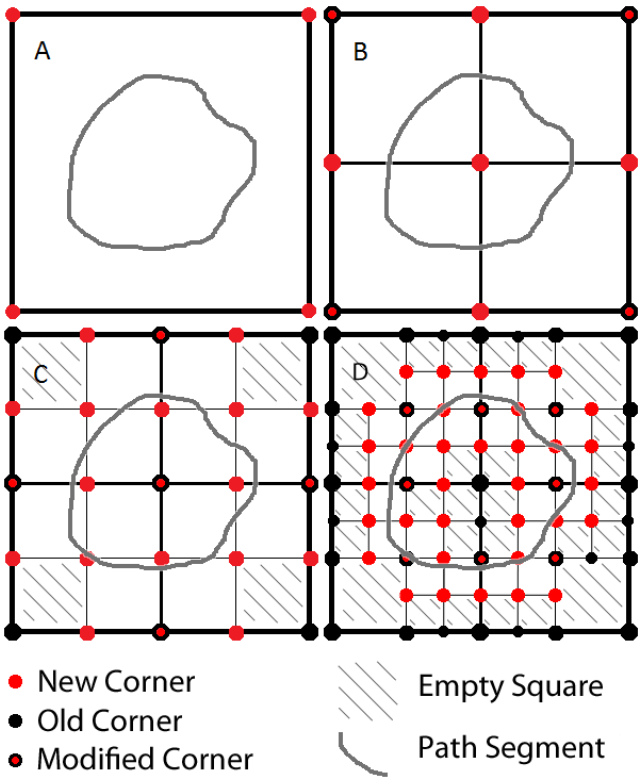

Figure 5: Four levels of the plane fitting method. Red circles indicate new points on the current quadtree level, while black circles show corners from previous levels. First the full plane is fitted using the full path, assigning four corner points (A). On the second level (B), 16 corner points are calculated, but several are averaged for continuity across segments. On the third level (C) we consider 16 squares, but not all squares contribute corner heights because the corner squares do not contain part of the terrain-path. In the fourth level (D), we do not need to refine the four corner squares because we already know they do not contain path points. Note that they can still change because some of the shared corners are still active.

We only apply the plane-fitting calculations on squares which contain enough points m from the terrain-path to satistfy the requirement from Section 4.2.1 that $m \geq n$. This creates a natural stopping point for our iteration. Once we have too few control points per square we will have an underdetermined system. For example, in a system solving for planes ($n = 3$) in a $1024x1024$ height map, we would be able to iterate 10 levels before we are down to $2x2$ squares, giv-

ing a maximum number of control points at $m = 4$. Further iterations would not allow for solving of a best-fit plane.

As in Figure 5C the corner squares (grayed out) of the terrain do not contain any part of the path, so they are not used to solve for a best-fit plane and no longer contribute to surrounding control corners. This does not mean that the square is held static, however. If that square is still adjacent to any other active squares, its control corners may still be affected, leading to a change in its shape.

4.3 Additional Elements

Apart from the shape of the terrain, we found that one of the biggest factors for immersion in the virtual environment were trees. Trees allow the subject to feel as if they are walking through the woods, rather than walking on a bare patch of grass. When moving, trees provide an easily perceived reference point for motion, enhancing the subject's immersive experience.

Our trees were modeled by hand but placed randomly through the terrain, although avoiding the path and area immediately surrounding the path. We found that placing enough trees in the environment allowed us to limit the size of terrain needed to surround the path, due to the visual occlusion provided by the trunks, branches, and leaves. The trees are maintained by Unity's terrain structure which provides support for instancing, billboarding, and distance-culling.

One more element which can be added to the environment is signs. These can provide the user with information such as the distance covered, the distance remaining, or warning them that a hill is just around the corner. They can also be used for encouraging messages from the operator, letting the user know he is doing a good job without removing him from the virtual world.

5. IMPLEMENTATION

We implemented the system using the Unity3D software package (http://www.unity3d.com). All scripting was done using C#.

Our basic scene begins with a set of control points for the path that the user can move, add to, or delete. The path can be toggled open or closed to form a loop. At any time the user can decide to build the terrain which begins the process described in Section 4. They can flatten the generated terrain to start again or edit the path which has already been built. Trees and signs can be added and deleted. The user has options to save the terrain they are currently working on or to load saved terrains. While modifying the path, a moving border informs the user of the anticipated size of the generated terrain.

Our scenes are able to run in real-time with interactive framerates over 90fps on a single core Intel i7 processor using under 750MB of memory for our largest terrain.

A part of the immersive system involves projecting the environment onto a domed surface to immerse the user in their environment. We developed a hemispherical projection surface built of foamcore on which to display the environment. With the projector behind the screen and a quarter-spherical front-surface mirror, we are able to show an image which encompasses the user's full field of view without encumbering motion.

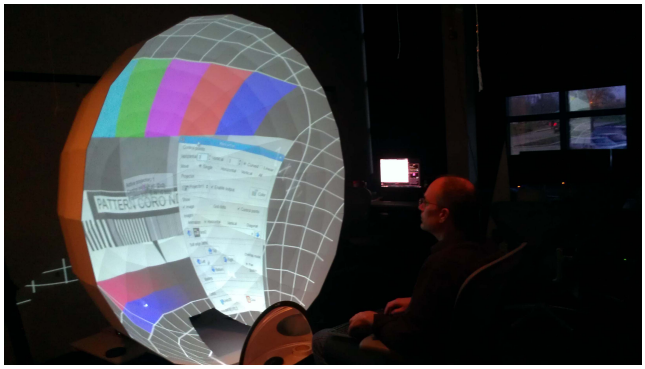

Figure 6: Testing the spherical projection system to properly register the image.

6. RESULTS

We have used this system to generate large terrains much more efficiently than can be done by hand. Our largest path created to date involved 250 control points and a path length of nearly 4km. The resulting terrain is over 1.5km square with a resolution of 4096^2, and takes around 30 seconds to iterate for its shape, paint the path texture, and populate with trees. We based this path off of a terrain modeled and textured by hand which took around 40 hours to design including shaping, smoothing, texturing, adding trees and other objects to the scene. The path itself, with 250 control points, took around a half an hour to complete. The resulting terrain shapes are comparable, however the time savings of 80x is very significant.

We also found that it is much easier to have a novice therapist figure out how to use the path creation tool than to learn all the modeling and texturing techniques required to build a terrain and path system from the ground up. This allows the system to be distributed to people with little 3D graphics experience.

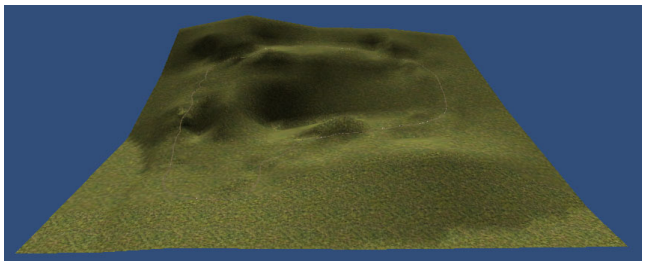

Figure 9: The terrain shape resulting from our large mountain path. The path length is nearly 4km, and the terrain is over 1.5km square with over 16 million points in the height map.

7. CONCLUSION

We have presented a system for generating environments based on user-defined paths. The path generation technique allows for novice users to define a path shape to fit the needs

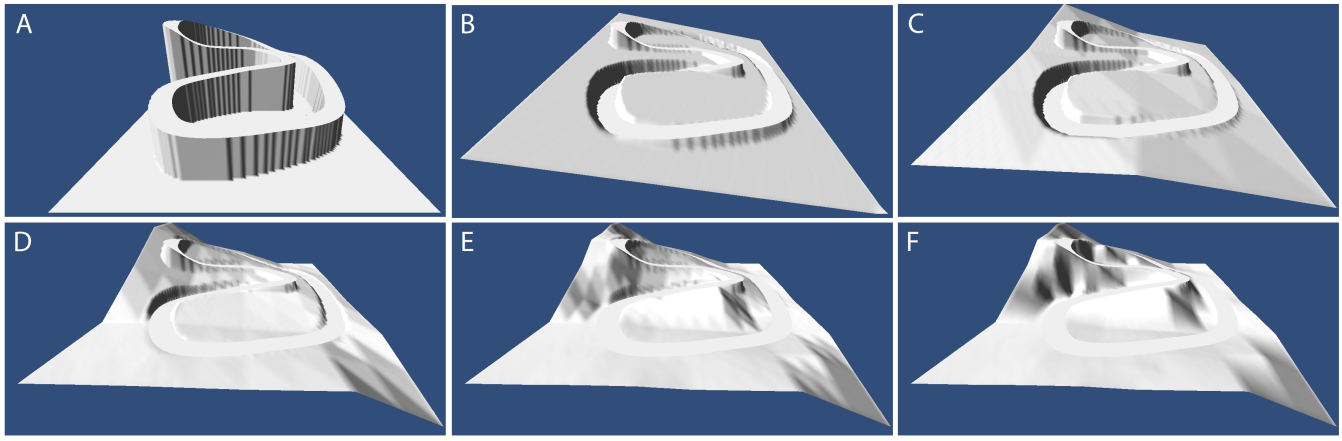

Figure 7: We begin the terrain fitting process by matching the terrain to the height of the path (A). Next we start our iterative planar matching algorithm. First we find the best fit plane for the full scale of the terrain (B). Next we iterate through different resolutions of planes and refine the fitting for each square (C-F) until the final terrain takes shape.

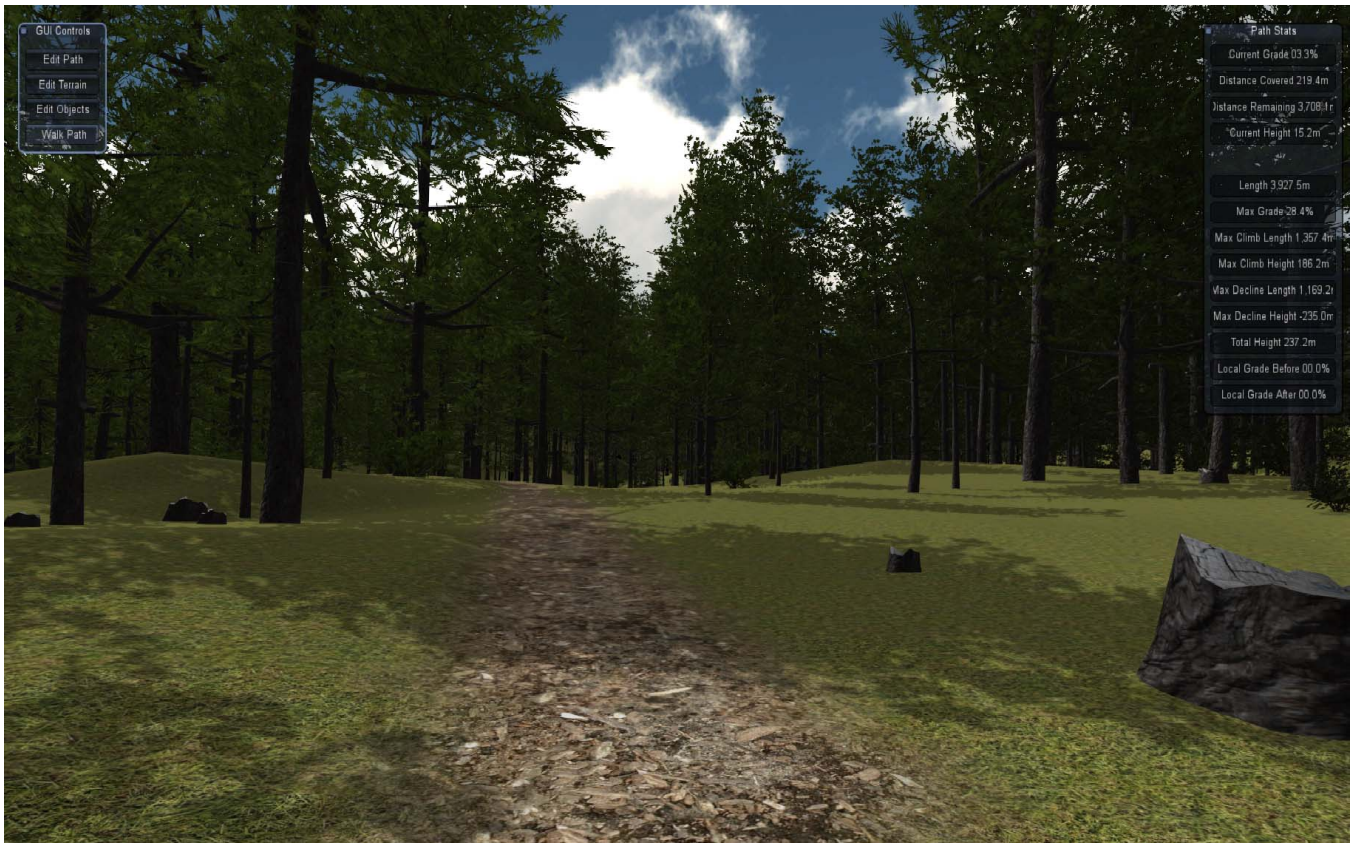

Figure 8: A screenshot of the environment while walking along the user-defined path. On the right are a list of statistics which can be used by the operator including the path length and slope.

of a physical therapy patient. The surrounding terrain is automatically generated to conform to the shape of the given path. Trees, signs, and textures are applied to the terrain to create an immersive experience for the user.

In future work we would like to extend the capabilities of the terrain system to include the ability to integrate hand-designed elements into the terrain shape. This could include features such as cliffs, creekbeds, lakes, and other physical formations. These features would enhance the realism of the experience for the user and could be applied to the scene in locations determined by image matching techniques or similar. We would also like to be able to provide interactive terrain shaping as control points are added and manipulated. We would also like to explore potential applications with more complex terrains including those with tunnels and multi-layered paths.

8. ACKNOWLEDGMENTS

I would like to thank Bertec for their funding and support of the project. Thanks to Chloe Shi for her helpful comments on the paper.

9. REFERENCES

[1] D. Adams, P. Egbert, and S. Brunner. Feature-based interactively sketched terrain. In *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, I3D '12, pages 208–208, New York, NY, USA, 2012. ACM.

[2] F. Belhadj. Terrain modeling: a constrained fractal model. In *Proceedings of the 5th international conference on Computer graphics, virtual reality, visualisation and interaction in Africa*, AFRIGRAPH '07, pages 197–204, New York, NY, USA, 2007. ACM.

[3] F. Belhadj and P. Audibert. Modeling landscapes with ridges and rivers: bottom up approach. In *Proceedings of the 3rd international conference on Computer graphics and interactive techniques in Australasia and South East Asia*, GRAPHITE '05, pages 447–450, New York, NY, USA, 2005. ACM.

[4] S. Berkovsky, M. Coombe, J. Freyne, D. Bhandari, and N. Baghaei. Physical activity motivating games: virtual rewards for real activity. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '10, pages 243–252, New York, NY, USA, 2010. ACM.

[5] F. Bevilacqua, C. T. Pozzer, and M. C. d'Ornellas. Charack: Tool for real-time generation of pseudo-infinite virtual worlds for 3d games. In *Proceedings of the 2009 VIII Brazilian Symposium on Games and Digital Entertainment*, SBGAMES '09, pages 111–120, Washington, DC, USA, 2009. IEEE Computer Society.

[6] N. Bianchi-Berthouze, W. W. Kim, and D. Patel. Does body movement engage you more in digital game play? and why? In *Proceedings of the 2nd international conference on Affective Computing and Intelligent Interaction*, ACII '07, pages 102–113, Berlin, Heidelberg, 2007. Springer-Verlag.

[7] E. Bruneton and F. Neyret. Real-time rendering and editing of vector-based terrains. *Computer Graphics Forum*, 27(2):311–320, 2008.

[8] H. Chandra, I. Oakley, and H. Silva. Designing to support prescribed home exercises: understanding the needs of physiotherapy patients. In *Proceedings of the 7th Nordic Conference on Human-Computer Interaction: Making Sense Through Design*, NordiCHI '12, pages 607–616, New York, NY, USA, 2012. ACM.

[9] D. Dimovska, P. Jarnfelt, S. Selvig, and G. N. Yannakakis. Towards procedural level generation for rehabilitation. In *Proceedings of the 2010 Workshop on Procedural Content Generation in Games*, PCGames '10, pages 7:1–7:4, New York, NY, USA, 2010. ACM.

[10] A. Fournier, D. Fussell, and L. Carpenter. Computer rendering of stochastic models. *Commun. ACM*, 25(6):371–384, June 1982.

[11] J. Gain, P. Marais, and W. Straßer. Terrain sketching. In *Proceedings of the 2009 symposium on Interactive 3D graphics and games*, I3D '09, pages 31–38, New York, NY, USA, 2009. ACM.

[12] E. Galin, A. Peytavie, N. Maréchal, and E. Guérin. Procedural Generation of Roads. *Computer Graphics Forum (Proceedings of Eurographics)*, 29(2):429–438, 2010.

[13] B. B. Mandelbrot. *The Fractal Geometry of Nature*. W. H. Freedman and Co., New York, 1983.

[14] F. K. Musgrave, C. E. Kolb, and R. S. Mace. The synthesis and rendering of eroded fractal terrains. In *Proceedings of the 16th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '89, pages 41–50, New York, NY, USA, 1989. ACM.

[15] K. Perlin. An image synthesizer. In *Proceedings of the 12th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '85, pages 287–296, New York, NY, USA, 1985. ACM.

[16] H. Samet. *The design and analysis of spatial data structures*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1990.

[17] J. Sinclair, P. Hingston, and M. Masek. Considerations for the design of exergames. In *Proceedings of the 5th international conference on Computer graphics and interactive techniques in Australia and Southeast Asia*, GRAPHITE '07, pages 289–295, New York, NY, USA, 2007. ACM.

[18] R. Smelik, T. Tutenel, K. J. de Kraker, and R. Bidarra. Integrating procedural generation and manual editing of virtual worlds. In *Proceedings of the 2010 Workshop on Procedural Content Generation in Games*, PCGames '10, pages 2:1–2:8, New York, NY, USA, 2010. ACM.

[19] S. Stachniak and W. Stuerzlinger. An algorithm for automated fractal terrain deformation. In *In Proceedings of Computer Graphics and Artificial Intelligence*, pages 64–76, 2005.

[20] J. Yim and T. C. N. Graham. Using games to increase exercise motivation. In *Proceedings of the 2007 conference on Future Play*, Future Play '07, pages 166–173, New York, NY, USA, 2007. ACM.

[21] H. Zhou, J. Sun, G. Turk, and J. M. Rehg. Terrain synthesis from digital elevation models. *IEEE Transactions on Visualization and Computer Graphics*, 13(4):834–848, July 2007.