Collision-free travel with terrain maps

Conference Paper · December 2009

DOI: 10.1145/1670252.1670289 · Source: DBLP

2 authors:

Andrei Sherstyuk
University of Hawai'i System
52 PUBLICATIONS **594** CITATIONS
SEE PROFILE

Anton Treskunov
30 PUBLICATIONS **220** CITATIONS
SEE PROFILE

Collision-free Travel with Terrain Maps

Andrei Sherstyuk*
Avatar Reality

Anton Treskunov †
Samsung

Abstract

Terrain maps, commonly used for updating elevation values of a moving object (i.e., a traveler), may be conveniently used for detecting and preventing collisions between the traveler and other objects on the scene. For that purpose, we project the geometry of all collidable objects onto the map and store it in a dedicated color channel. Combined with adaptive speed control, this information provides fast and reliable collision-avoidance during travel, independent of scene complexity. We present implementation details of the base system for a Virtual Reality application and discuss a number of extensions.

CR Categories: I.3.6 [Computer Graphics]: Methodology and Techniques—Interaction techniques;

Keywords: Collision avoidance, virtual travel, auto-speed control.

1 Introduction

Navigation is one of the most fundamental tasks, both in real and virtual environments. A successful navigation system must provide task-oriented travel metaphors, intuitive user interface, and unobtrusive and efficient control mechanisms for helping users when they find themselves in difficult travel situations [2001; 2005].

Collision prevention system is one of the most important parts of that control mechanism. It must be able to detect all pending collisions and take measures to prevent them. Both tasks are difficult in the general case of dynamically changing environments, when the collider and collidees are moving freely. A number of existing algorithms for processing collisions are discussed in [Akenine-Möller et al. 2008] and [Jimnez et al. 2000]. One of the more recent techniques for collision detection and avoidance is presented by [Sakr and Sudama 2008]. Their method is specifically designed for fast-action massively multiplayer gaming environments, with a very large number of avatars, flying in 3D in unconstrained fashion.

However, in many applications, the collision detection problem may be reduced from 3D to 2D case, if travel is confined to a terrain surface. Furthermore, if the environment remains static, geometry boundaries of all potential colliding objects may be conveniently embedded into the terrain image map. Thus, at any time during travel, all upcoming collisions may be detected at a flat cost of projecting the current user location onto the terrain map and reading the pixel values in the area of interest. This information may also be used by the system to help traveler avoid an upcoming collision.

In this work, we describe the entire system in detail. It is fast, reliable and allows to navigate around obstacles of arbitrary shapes. The system was implemented and successfully used in a number of medical Virtual Reality (VR) applications.

*e-mail: andrei@avatar-reality.com
†e-mail:anton.t@sisa.samsung.com; work performed while at USC.

2 Motivation and Application Requirements

The navigation system presented below was initially developed for a VR application for pain control and biofeedback research at Tripler Army Medical Center in Honolulu, Hawaii [2005]. Later, this system was rebuilt for simulating search-and-rescue training missions in VR. Also it was used for teaching medical students triage skills and life saving procedures [2008]. In all configurations, users had to navigate on uneven terrain in outdoors settings, avoiding collisions with natural obstacles and human artifacts. The areas available for travel were typically small (under a few hundred square meters) and all obstacles were predominantly static: trees, buildings, rocks, etc. One of the scenes is shown in Figure 1.

For these applications, using a raster image for storing terrain elevation map was a natural choice. The images were generated procedurally from a set of regular polygonal meshes that represented the terrain surface. The highest and lowest vertices on the surface corresponded to white and black pixels in the image map, respectively. The 3D meshes were obtained from satellite datasets of existing locations on Hawaii.

Figure 1: *A small sandy island with rock formations and stylized Stone Age dwellings, used for search-and-rescue missions in VR.*

3 Generating Collision Map

In order to incorporate colliding objects into the map, the following steps were taken. First, the terrain mesh and all colliding objects were loaded into Maya and positioned as needed on scene. All non-terrain objects were grouped and translated vertically above a predefined threshold. Non-colliding parts of objects, such as building roofs and pillar tops, were removed. Then, colliding objects were scaled down vertically (flattened), to produce 2D areas forbidden for travel. The scene was exported into a single 3D mesh of polygons in OBJ format, displayed in Figure 2, top. The mesh was post-processed converting the elevation data for all vertices into a TIFF RGB image, with 8 bits per channel. Vertices with elevation above the threshold value were output as red pixels, indicating zones forbidden for travel. The resulting sparsely spaced pixels were inflated towards the nearest neighbors, filling the gaps

between samples. The grayscale pixels with valid elevation data were added using interpolation; the red pixels were propagated as is, keeping their values unmodified. Finally, the TIFF elevation map with red no-travel zones was hand-edited in order to remove noise and make sure that objects' borders are continuous. The annotated resulting map is shown in Figure 2, bottom.

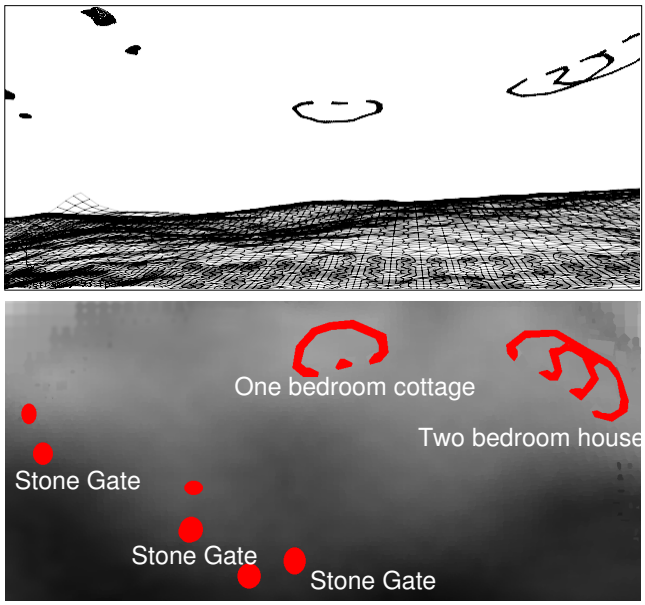


Figure 2: *Top: a terrain mesh with flattened colliding objects, placed high in the air. The roofs are removed from the cottage and the house. Bottom: complete terrain map for the island scene with no-travel zones painted in red – the house and the cottage walls and the pillars supporting massive stone gates. Area size: 30 x 60 meters, image size 600 x 1200 pixels, resolution 5 cm/pixel.*

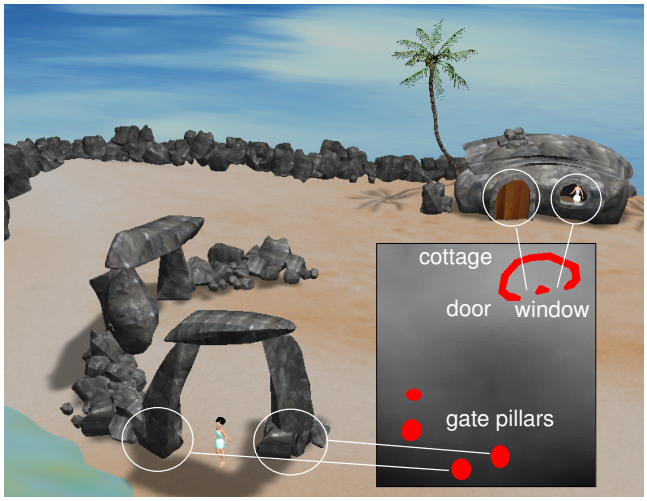


Figure 3: *Traveling on an island scene. On the terrain map fragment (bottom right), no-travel zones are painted in red. Notice that the cottage may be entered via the door or the window. Virtual travelers can go over small rocks, but not through the pillars that support massive gates.*

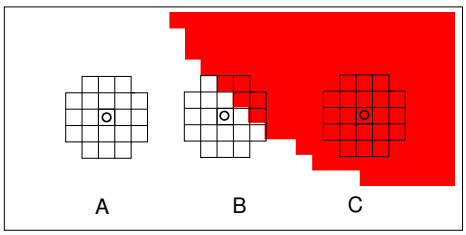
4 Collision-Avoidance Algorithm

During travel, the system continuously updates the user location, by adding incremental displacements on the (X, Z) plane according to the current velocity values. The vertical displacement is obtained by bilinear interpolation between the four corners of a heightfield image map, at pixels $(i, j), (i + 1, j), (j + 1, j + 1), (i, j + 1)$. Here, (i, j) are pixel coordinates on the image map, corresponding to the user location on (X, Z) plane. The map also contains additional areas that are prohibited for travel, such as walls, tall rocks, as explained in the previous section. These areas are painted in red.

Each time when the user moves to the next pixel, the system calculates the "danger level" (D), proportional to the number of red pixels in the vicinity of the new location. Pseudo-code for computing D-value is shown in listing 1. The $radius$ parameter defines the size of the search area. It is measured in pixels. In this example, $radius$ is set to 2 pixels, covering the area of 5 x 5 pixels, minus 4 corner pixels in order to make the search area symmetrical in all directions. Total of 21 pixels are checked for red values. Thus, the granularity of $D(radius = 2)$ is 1/21. Increasing the radius makes D smoother. Danger level ranges from $D = 0$ (no red pixels, no restrictions on travel) to $D = 1$ (max danger, travel impossible).

Once the D value is obtained, the speed control mechanism is applied: $S = S_0\ (1 - D)^2$, where S_0 is the requested speed as set by the user, and S is the actual speed, allowed by the travel system. If the user persist in moving deeper into the red zone, (D-value increases), the system will eventually bring him or her to a full stop, followed by a forced relocation to the last memorized location, where D had zero value. If the user chooses to move outside the red zone, the system encourages this decision by restoring the speed to initial value S_0. Thus, it is easier for users to move outside of dangerous areas, than to travel deeper inside. The speed control mechanism is presented in listing 2.

Algorithm 1 *Calculations of the danger level are performed on each cycle of the graphics loop, before updating the traveler's speed and position. In the diagram, the user is moving towards a gate pillar. The danger level changes as follows: 0 (A), 5/21 (B), 1 (C). At point C, the user is stopped and relocated to the last safe location, point A.*




```
01: procedure danger_level(Image map, Pixel point, radius R)
02:   N_red ← 0
03:   N_total ← 0
04:   for all pixels p : ||p − point|| ≤ R do
05:         N_total ← N_total + 1
06:         Color RGB ← get_pixel_value(map, p)
07:         if R = 1 and G = 0 and B = 0 then
08:               N_red ← N_red + 1
09:         endif
10:   endfor
11:   danger_level ← N_red/N_total
12: end procedure
```

Algorithm 2 *Speed control mechanism updates the current value of danger level D and, if it increases, reduces the speed (Line 11). In critical cases, when the danger level reaches max value of 1, user comes to a full stop (Line 8).*

```
01: procedure update_speed(Vector position, Image map)
02:   Pixel p ← world_to_map(position, map)
03:   D ← danger_level(map, p, radius = 2)
04:   if D = 0 then
05:         save current position
06:         factor ← 1
07:   else if D = 1 then
08:         go back to saved position
09:         factor ← 0
10:   else if D > D_previous then
11:         factor ← (1 − D)^2
12:   else
13:         factor ← 1
14:   endif
15:   speed ← speed_0 * factor
16:   D_previous ← D
17: end procedure
```

5 Implementation and Results

The described travel control system was implemented in *Beach-World*, a medical VR application [Sherstyuk et al. 2005] and in *VR-Triage*, a Virtual Reality trainer for first responders [Vincent et al. 2008]. Both applications were built using [Flatland 2002] open source engine. These systems were used for a number of experimental studies, including technical VR evaluation, medical education and clinical research. One of these experiments is shown in Figure 4. The user is navigating on the island scene, wearing a headset (5DT800 HMD, 800 x 600 pixels, 40° viewing field) and magnetic sensors for head and hand tracking. We used Flock of Birds tracker by Ascension, running in 9 feet extended range.

We collected data from 13 participants (adults between 30 and 50 years of age, both male and female) who volunteered to test *VR-Triage* system at the Medicine Meets Virtual Reality conference. For this particular exercise, all participants were asked to enter the village by passing between the gates and visit every room in every building, searching for virtual patients. Details relevant to navigation part of their experience are summarized below:

	Min	Median	Mean	Max
Time on mission (sec)	132	348	380	687
Time spent traveling (sec)	29	72	82	161
Travel distance (meters)	86	187	250	649
Number of forced stops	0	3	3.16	8

As the table indicates, people had between 0 and 8 forced stops, when they moved too deep into no-travel areas and were stopped by the collision-avoidance system. Each detected stop, followed by a forced relocation to saved position, was time-stamped and recorded into a travel log file. The plot in Figure 5 displays the number of stops per person over time, averaged over one minute interval. Visual observations show that the actual collisions did not happen at all, because the virtual camera did not penetrate the obstacles: building walls and stone pillars. Personal observations also showed that very few people had problems with the speed control mechanism. None of them reported travel difficulties or motion-sickness in the follow-up survey. All participants were able to navigate successfully in all areas, including the two-bedroom house (north-east corner of the map on Figure 2), with irregular narrow elongated rooms.

Figure 4: *A virtual traveler is negotiating his way between large stone pillars, using steering-by-pointing technique. The first person view is also projected onto a screen (top right).*

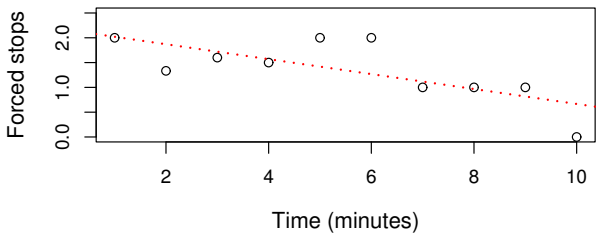


Figure 5: *Number of forced stops per person, over time. Travelers showed the tendency to reduce the number of forced stops, from 2 to 1, as they progressed with their task. A linear model was fit, plotted as a dotted line, with $p_{slope} = 0.01738$. Total number of samples 38, for all 13 participants.*

6 Extensions

Besides indicating no-travel zones, terrain image maps may also store other useful information, related to the current position of the traveler. For example, an image map can be used for location-based lighting control. In Figure 6, a traveler approaches the cottage and enters it through the window. While the traveler remains outside, the scene is illuminated by a single directional light. When the traveler moves inside, this light is dimmed and a local head-mounted spotlight is activated, to simulate indoors lighting. This location-based change of lighting may be conveniently controlled by one of color channels in the terrain map. For outdoors areas, all pixels in the map will have conventional grayscale values. For all indoors areas, one dedicated channel (e.g., blue) must be set to zero or any other predefined value. Unlike red pixels $(255, 0, 0)$, used for marking no-travel zones, pixels that control lighting contain a full range of red and green values, from $(255, 255, 0)$ to $(1, 1, 0)$. Therefore, pixels with modified blue channel can still be used to update the user elevation, by taking the values from unmodified channels.

Another application of selective use of color channels is setting

Figure 6: *The travel system can automatically select between outdoors lighting (left, directional sunlight) and indoors lighting (right, head-mounted spotlight), by checking values in the blue channel of the terrain map. All pixels in elevation map inside the building have blue values set to 0.*

zones prohibited for teleportation. In *VR-Triage* system, users are offered two travel metaphors: continuous steering-by-pointing and fast automatic relocation towards a selected target, or teleportation for short. When by steering, users move at variable speed, from 3 to 15 km/hour. During teleportation, people move at fixed speed of 20 km/hour. Experimental evidence shows that teleportation is very popular, especially among novice users, because it allows to minimize travel efforts and guarantees arrival at a desired destination point. However, in certain locations, teleportation should not be used. For example, entering and leaving buildings should be done by low-speed maneuvering. In these high-collision areas, teleportation commands may easily results in attempts to move through the walls, closed doors, window and door frames. Automatic speed control will inevitably stop these moves, which may cause much frustration from users. Restrictions on teleportation may be applied to incoming and outgoing travel requests, separately. "No-teleportation" zones can be painted onto the map, by setting one of the color channels to some predefined value.

In all these examples, the additional information stored with the terrain map has bitwise nature – it turns certain features on and off. By applying masks to two 8-bit wide color channels, one can have practically unlimited number of bitwise combinations for customized location-based maps. (Note, that one color channel should be reserved for the terrain elevation data.) Modified color channels can store range data, too. That extension can be useful, for example, for soundscaping the scene. The blue channel can be assigned to control the volume of the ocean sounds, painted along the shoreline on the island map. The green channel may be used to set locations of tropical birds or other inland sounds. In both cases, a simple image editor may be used for fast prototyping of locations and intensity of various sounds.

7 Conclusion

We described a system for collision-free travel in VR, that is suitable for applications, where scene layout is static and heightfield maps can be applied. The proposed algorithm of speed control is easy to incorporate into most travel techniques. We have implemented it for steering-by-pointing, steering-by-joystick and for target-based auto-relocation (teleportation).

The novelty of this work lies in the idea of treating separate color channels in terrain maps independently. In a general case, RGB color triplets, stored in the terrain image, can be modified and used for a variety of purposes, while still remaining functional as a height-field raster data array. Use of color image maps allows easy and quick prototyping of 3D scenes, by simply painting these features onto the map. We believe that proposed extensions of traditional terrain maps will be useful in a variety of interactive real-time 3D applications.

References

AKENINE-MÖLLER, T., HAINES, E., AND HOFFMAN, N. 2008. *Real-Time Rendering 3rd Edition*. A. K. Peters, Ltd., Natick, MA, USA.

BOWMAN, D. A., KRUIJFF, E., JR., J. J. L., AND POUPYREV, I. 2001. An introduction to 3d user interface design. *Presence 10*, 1, 96–108.

FLATLAND. 2002. *The Homunculus Project at the Albuquerque High Performance Computing Center (AHPCC)*. http://www.hpc.unm.edu/homunculus.

JIMNEZ, P., THOMAS, F., AND TORRAS, C. 2000. 3d collision detection: A survey. *Computers and Graphics 25*, 269–285.

RUDDLE, R. A. 2005. Navigation: Theoretical foundations and techniques for travel. *Virtual Reality Conference, IEEE 0*, 311.

SAKR, Z., AND SUDAMA, C. 2008. A curvilinear collision avoidance scheme for interactive 3d gaming environments. In *ACE '08: Proceedings of the 2008 International Conference on Advances in Computer Entertainment Technology*, ACM, New York, NY, USA, 130–133.

SHERSTYUK, A., ASCHWANDEN, C., AND SAIKI, S. 2005. Affordable virtual environments: Building a virtual beach for clinical use. In *13th Annual Medicine Meets Virtual Reality Conference*, IOS Press.

VINCENT, D., SHERSTYUK, A., BURGESS, L., AND CONNOLLY, K. 2008. Teaching mass casualty triage skills using immersive three-dimensional Virtual Reality. *Academic Emergency Medicine 15*, 11, 1160–5.