International Conference on Artificial Reality and Telexistence
Eurographics Symposium on Virtual Environments (2014)
T. Nojima, D. Reiners, and O. Staadt (Editors)

Space-Time Maps for Virtual Environments

Andrei Sherstyuk[1] and Anton Treskunov [2]

[1] University of Hawaii, USA
[2] Samsung, USA

Abstract

Terrain image maps are widely used in 3D Virtual Environments, including games, online social worlds, and Virtual Reality systems, for controlling elevation of ground-bound travelers and other moving objects. By making use of all available color channels in the terrain image, it is possible to encode important information related to travel, such as presence of obstacles, directly into the image. This information can be retrieved in real time, for collision detection and avoidance, at flat cost of accessing pixel values from the image memory.
We take this idea of overloading terrain maps even further and introduce time maps, where pixels can also define the rate of time, for each player at given location. In this concept work, we present a general mechanism of encoding the rate of time into a terrain image and discuss a number of applications that may benefit from making time rate location specific. Also, we offer some insights how such space-time maps can be integrated into existing game engines.

Categories and Subject Descriptors (according to ACM CCS): H.5.1 [Computer Graphics]: Multimedia Information Systems—Artificial, augmented, and virtual realities

1. Introduction

2D terrain maps have been used for creating realistic landscapes in 3D environments for nearly three decades [Mil86]. Besides providing polygonal render meshes of the surface, terrain maps enable direct control over elevation of all objects moving on the terrain. Such objects include user avatars, non-player characters, ground vehicles, and other entities. Besides elevation values, terrain maps may provide other location-specific information, such as slope angle and surface material properties.

For convenience, most 3D graphics engines and authoring tools allow to create terrain maps by importing pre-existing images, for example, synthetic fractal clouds, as the first draft of the terrain that can be refined as desired. Terrain maps can also be exported as graphics images, for sharing and assets packing purposes. In order to provide high precision in calculating object elevation, such images are stored with 16-bits per pixel depth. For example, Unity 3D provides the option to import and export terrain maps in 16-bit RAW image format, compatible with most image editors.

The internal representation of terrain maps varies between systems, from simple texture buffers, used in Flatland engine [Fla] to complicated compiled data structures that support multiple level of surface detail in CryEngine [CT]. To conserve run-time memory, most engines strip extra color channels from RGB images when importing terrain images, treating them as monochrome pixels. However, some engines use the original image, which can be in RGB or even in RGBA pixel formats. That opens many interesting opportunities for creative use of additional color channels.

Recently, it was shown how dedicated colors in terrain maps (e.g., pure red pixels) can be used for detecting and processing collisions with static and dynamic obstacles, in real time [SK14]. The authors also suggested that unused color channels in terrain maps may store extra scene information, such as layouts and intensities of spatially distributed sound and light sources. By doing so, every 3D scene may be "soundscaped" or "lightscaped", by painting them in Photoshop, where hue value of the brush will define a source type, and saturation will define its intensity.

In this concept work, we further explore the idea of encoding location-specific data into terrain maps. Namely, we introduce a direct mapping between space and time, making the time rate a local property of space.

DOI: 10.2312/ve.20141363

2. Time Maps for Virtual Worlds

Consider a test scene in Figures 1 and 2, showing some virtual minimalistic Square City, surrounded by mountains. The monochrome terrain map has the elevation data stored as (r,g,b) float values, where all three color channels have equal values $r = g = b$, ranging from 0 to 1, for every pixel on the map. The pedestrian crossings, however, have $g > r$, which makes these pixels look green. The scene was staged and rendered in Flatland 3D engine [Fla].

We suggest to use g-values for controlling time increments in all internal clocks and other time-keeping structures, for all objects on the scene. In this example, the walking human character is animated with motion capture data (MOCAP), repeating the walking cycle. If the update rate is set to 30 FPS, the character's joint angles are fetched and re-sampled from MOCAP data arrays with the constant time increment $\Delta t = 1/30$ second. In the proposed system, Δt is modified by a location-specific function $f(r,g,b)$:

$$\Delta t = f(r,g,b)\ \Delta t_0 \tag{1}$$

where Δt_0 is the inverse frequency of the simulation

$$\Delta t_0 = 1/FPS \tag{2}$$

and transfer function f may be defined as

$$f(r,g,b) = \begin{bmatrix} 1 & mono\ pixels, r = g = b \\ T(2g-1) & color\ pixels \end{bmatrix} \tag{3}$$

In this form, the values of the green channel completely define how time increments are computed. Time may be accelerated to some maximum value T ($g = 1, f = 1$), frozen ($g = 0.5, f = 0$) and even reversed ($g < 0.5, f < 0$). All these effects may be achieved by painting pixels on the terrain map with a desired shade of green. Note, that the terrain map remains fully functional for computing elevation, by accessing values from other channels.

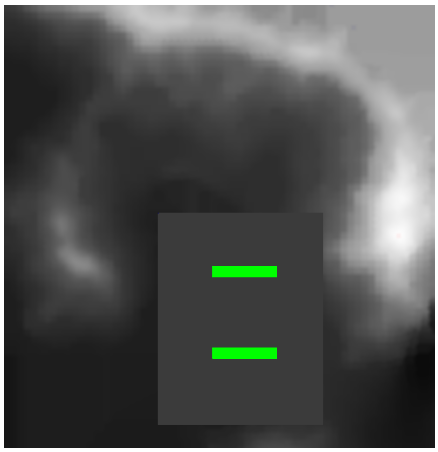

Figure 1: *Terrain map of Square City, set inside a volcanic crater. Green pixels define places in the scene where time runs faster. These locations correspond to two zebra crossings, shown in rendered image in Figure 2.*

Figure 2: *"Fast-forward" zebra crossing in Square City. When the walking character steps on green-pixel area, his internal time advances using larger increments, making him move faster.*

3. Related Work

In the test scene, shown in Figures 1 and 2, the time map contains two accelerated zebra-crossings that are meant to help organize pedestrian traffic. Thus, when used as a travel aid, time maps bear certain resemblance to navigation meshes, that define areas on the scene where travel is possible [Sno00]. Navigation meshes were adopted by many game engines, including CryEngine, Unity 3D and Unreal, for pathfinding purposes. Initially created by hands, these meshes are now generated automatically, with various levels of optimizations and quality control [KK14]. One of the recent methods involves initial voxelization of the scene, followed by extraction of walkable levels [OP13].

Similarly to navigation meshes, the proposed time maps can be viewed as means of quantifying the entire navigable space by maximal allowed travel speed up to T value (see equation 3). However, time maps are different in three important aspects:

- the layout and intensity of zones with modified time rate are not bound by local geometry;
- the resolution of space partitioning is limited only by the size of the terrain map, and can be set to be arbitrarily fine;
- applications of time maps are not limited to aiding travel.

Because of these properties, time maps are able to produce effects that can not be achieved by other methods. These will be discussed next.

4. Using Time Maps in Virtual Worlds

We loosely grouped several suggested applications into two categories: realistic and creative.

4.1. Better Travel and Urban Planning: Realistic Use

Applications from the first category mostly aim to facilitate navigation in realistic or slightly futuristic urban scenes. Zebra crossings, escalators, moving sidewalks and walkways, as used in airports – these are examples how pedestrian traffic can be improved by faster-than-life time flow.

Continuous modes of travel, such as walking or running, are often more preferable to instantaneous teleportation, because they do no disrupt the sense of presence, especially in immersive Virtual Reality (VR) settings. In some online social worlds, for example, Blue Mars [BM], teleportations are deliberately avoided, for the same reason. Limited use of teleportation also helps minimize avatars popping in and out in busy locations.

To help players get around faster, Blue Mars offered a number of traveling devices, such as scooters, for outdoor travel, and escalators for moving indoors, as shown in Figure 3. However, motorized vehicles are not available at all locations. In addition, avatars can only walk but not run on sloped surfaces, including staircases and escalators. Thus, accelerated walking and running in selected areas may be very effective. Because of large size of cities on Blue Mars (4 sq. km), moving in fast-forward mode could save players hours of tedious walking.

Similar issues exist in SimCity, a popular multiplayer urban simulator by Maxis [Sim]. In the latest version 4, the limit in city size was increased from 4 to 16 sq. km, due to demand from developers and general public.

Figure 3: *Multi-level shopping mall in Beach City on Blue Mars, connected with long escalators and staircases. In Blue Mars, running is not allowed on sloped surfaces, thus, walking in fast-forward mode could be particularly helpful.*

4.2. "Places Where Time Stood Still": Creative Use

It is easy to imagine how time maps can be employed in less realistic worlds, where everything goes to enhance user experience and game-play. Two cases are discussed below.

- *Time Alterations in Social Settings*
  Certain locations may be painted with dark green pixels, making players move in slow-motion. Such mass-induced behavior will be appropriate in virtual temples, shrines, mausoleums and various ceremonial sites where visitors are expected to observe certain rules. Conversely, a little speed-up of time rate could be welcome in more dynamic settings, such as discotheques, sports bars and other places of active recreation.
- *Green Pixels of Altered Time as User Resource*
  In all previous examples, areas with altered time rate are assumed to be permanently painted onto the map by a scene designer. Alternatively, green pixels may by placed and removed from the map by players themselves. Thus, time-modifying pixels become a valuable game resource, that players can mine in game or buy as a commodity.

As an example, consider Minecraft, "a game about breaking and placing blocks" [Min]. Players mine for resources and use them to create their worlds. When playing on public servers, great efforts are spent on building protection around homes and bases, where players keep their possessions, from raiders. Imagine if players could mine green pixels (or blocks, in Minecraft terms) of altered time in the same way they mine for ore. By placing dark-green pixels on the map, they could surround their homes with invisible perimeters, where time runs very slowly or even changes direction. Trespassers will be "glued" or deflected from the perimeter, unless they learn how to jump over the dark-green pixels, or make a passage by placing counter-acting bright-green time-accelerating pixels.

Another use for time-accelerating pixels is farming, a popular activity in Minecraft. Growing crops takes time, and that time can be reduced by placing bright-green pixels on the field.

5. Implementation

As was already mentioned, internal representation of terrain maps varies greatly in existing 3D engines. Below, we briefly compare implementation cost of time maps in three systems, from lightweight Flatland to photorealistic CryEngine.

5.1. Flatland

Flatland is an open-source 3D engine that was developed at Albuquerque High Performance Computing Center (AHPCC) to serve as a platform for research projects in VR [Fla]. Due to its simplicity and modular design, Flatland found many uses outside of AHPCC, mostly in the fields of medical VR simulation and 3D user interface design.

Flatland has the most straightforward implementation of terrain maps: they are based on conventional textures, with RGBA color channels. Thus, the system is fully ready for time maps, in its current state.

5.2. Ogre

Ogre is a popular open-source engine that was launched in 2001 and still remains in active development [Ogr]. Ogre was among the first public engines that offered support for stereo rendering, which made it a platform of choice for many projects in VR.

In this system, terrain images are stored as monochrome float values, with no room for extra channels. However, Ogre allows to load and store an optional source image for terrain, that can have pixels defined in variety of formats, including RGB and RGBA. In addition, Ogre provides converters between the source image and internal terrain map representation, which makes the system compliant with the time maps requirements.

5.3. CryEngine

CryEngine series was developed by CryTek GmbH [CT] and was intended primarily as a licensed high-end photorealistic platform for commercial 3D games. However, over the past years, CryTek released most of the code and started to provide support for developers. Thanks to recently added support for stereoscopic display devices, CryEngine is now actively used by academic community for research in VR [BIPS12].

As Ogre, Unity 3D, and many other engines, CryEngine stores terrain images as arrays of floats, with a single float value per pixel. Unlike Ogre, though, CryEngine does not keep the source terrain image at run-time. Fortunately, many operations on terrain images are templated in the code. In Sandbox, the level editor for CryEngine, terrain maps are defined as

```
typedef TImage<float> CFloatImage;
```

which makes expanding monochrome pixels to RGB a trivial task. In the real-time game code, this task is more complicated, but still possible due to consistent use of templates that allow reading data from image files in desired pixel formats. However, rather than modifying existing terrain data structures, a simpler and less-intrusive solution would be implementing an additional RGB image for storing the time map, similar to Ogre's source image.

Summary: time maps can be implemented in modern game engines with little effort.

6. Limitations and Future Work

Being a concept work, this paper offers simplified formulas for computing time increment. An obvious problem arises when painted green time pixels happen to have the same value as red terrain pixels. This limitation ought to be addressed in the future work.

Such future work may have a complete implementation of proposed time maps in one of aforementioned game engines, so the evaluation of proposed technique could be performed.

7. Conclusion

We presented a novel extension of conventional terrain image maps. Our approach is based on encoding the rate of time into a dedicated color channel of the map. The resulting time maps enable explicit control over all time-related processes for all entities in the virtual world, with the spatial resolution of the terrain map. The proposed system may be added to any 3D environment, where conventional terrain maps are used. Most importantly, time maps provide a multitude of exciting ways to control the virtual environment and all its content.

References

[BIPS12] BRUDER G., INTERRANTE V., PHILLIPS L., STEINICKE F.: Redirecting walking and driving for natural navigation in immersive virtual environments. *IEEE Transactions on Visualization and Computer Graphics 18*, 4 (2012), 538–545. 4

[BM] Blue Mars Online. http://www.bluemars.com. 3

[CT] CryEngine, Crytek. http://mycryengine.com. 1, 4

[Fla] Flatland, The Homunculus Project at the Albuquerque High Performance Computing Center AHPCC. http://www.hpc.unm.edu/homunculus. 1, 2, 3

[KK14] KALLMANN M., KAPADIA M.: Navigation meshes and real-time dynamic planning for virtual worlds. In *ACM SIGGRAPH 2014 Courses* (New York, NY, USA, 2014), ACM, pp. 3:1–3:81. URL: http://doi.acm.org/10.1145/2614028.2615399, doi:10.1145/2614028.2615399. 2

[Mil86] MILLER G. S. P.: The definition and rendering of terrain maps. In *SIGGRAPH '86* (1986), ACM, pp. 39–48. 1

[Min] Minecraft. http://minecraft.net. 3

[Ogr] Ogre Open Source 3D Graphics Engine. http://www.ogre3d.org/. 4

[OP13] OLIVA R., PELECHANO N.: Neogen: Near optimal generator of navigation meshes for 3d multi-layered environments. *Computer & Graphics 37*, 5 (2013), 403–412. 2

[Sim] SimCity Official Website. http://www.simcity.com/. 3

[SK14] SHERSTYUK A., KIYOKAWA K.: Collision-free navigation with extended terrain maps. In *Transactions on Computational Science XXIII*, Gavrilova M., Tan C. J. K., Mao X., Hong L., (Eds.), vol. 8490. Springer Berlin Heidelberg, 2014, pp. 139–156. doi:10.1007/978-3-662-43790-2_8. 1

[Sno00] SNOOK G.: Simplified 3d movement and pathfinding using navigation meshes. In *Game Programming Gems*, DeLoura M., (Ed.). Charles River Media, 2000, pp. 288–304. 2