High-Detail Real-time Deformable Terrain on the GPU

Justin Crause[*]
Department of Computer Science
University of Cape Town

Andrew Flower[†]
Department of Computer Science
University of Cape Town

Peter Juritz[‡]
Department of Computer Science
University of Cape Town

Abstract

This project will investigate some implementations that have been proposed to tackle terrain deformations in real-time. This area of research has particular impact in 3D games and virtual environments which could greatly benefit from such techniques. Two different deformation techniques will be investigated and evaluated to find the best solution to this problem. A software package will be developed to study these two methods by displaying the environment and allowing the user to apply deformations. This package will include two control mechanisms for navigating and applying deformations to the environment. The purpose of this is to compare the effectiveness of each of these control systems.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism

Keywords: terrain deformation, bump mapping, tessellation, computer vision, displacement mapping

1 Problem Statement

Graphics based applications such as computer games and animated films have ever-increasing requirements. A large number of these applications require the display of virtual terrains and up until recently the majority of these terrain systems have been restricted to static geometry. In order to implement deformable and dynamic terrains new techniques must be developed that can produce visually appealing results and are furthermore able to perform under the time constraints that many of these applications include. These results need to be achievable in real-time in order to be of use in gaming and visualisation applications. Previous work on this topic focused on techniques to produce visually appealing results but due to their complexity could not run in real-time. With recent advances in computer graphics hardware these goals are becoming realisable. By taking advantage of the immense processing power that these devices offer it will be shown that a program can be created to run in real-time.

The goal of this project is to create a terrain deformation test-bed that supports the real-time deformation of a triangle mesh representing a terrain. The system must be capable of handling a large number of deformations whilst maintaining real-time frame-rates. Ultimately this system will serve as a precursor to further work on realistically deforming sandy terrains for computer games and visual effects. While methods do exist to change geometry on the fly, these have not been widely adopted in computer games mainly due to previous limits imposed by graphics hardware. Currently most computer games rely on precomputed data, meaning that the terrain (or other environmental objects) can only be changed in a small number of predefined ways, thus breaking immersion.

This project aims to evaluate methods to achieve these goals which do not require precomputed information and can run on mainstream graphics hardware. To this end, the necessary trade-offs in speed and representational error will need to be carefully analysed. Two deformation implementations will be examined during this project, one utilising geometry shaders to add new geometry in an unbounded approach and the other making use of textures to achieve similar visual effects but in a resource-limited fashion. The geometry shader approach will add new vertex data to the surface mesh to add in detail and deformations. After a large number of deformations are applied the number of vertices to process becomes too large. At this point the system would not be running with real-time frame-rates. A possible solution to this is to utilise texture techniques like bump and displacement mapping to achieve similar results but be able to handle a large number of deformations. In order to address these issues, a number of algorithms and techniques will be designed, implemented and evaluated with respect to how well they solve the frame-rate problem whilst still maintaining realistic appearance. There needs to be some method to interact with the deformation systems in an easy way. This will be addressed through the design of two separate interfaces, the first is a simple point-and-click interface. The second interface will be a computer vision based system wherein the user will use their hands in gesture motions to directly control the program.

[*]e-mail: jcrause@cs.uct.ac.za
[†]e-mail: aflower@cs.uct.ac.za
[‡]e-mail: pjuritz@cs.uct.ac.za

2 Procedures and Methods

The proposed implementation of a real-time deformable terrain system aims to provide two alternatives to representing the deformation. These techniques will be performed on the GPU and will be controllable in real-time by the user. Two methods of interaction are proposed: a point-and-click mouse-orientated interface and a computer vision interface based on hand tracking. These methods are discussed in more detail in the sections below.

2.1 Interface Design

The implementation of a point-and-click interface is planned, allowing users to navigate the environment as well as apply deformations to the terrain. This interface will render the environment in 3 dimensions. The user will navigate the environment by flying around in the environment space. Terrain deformations will then be made by selection an area with the pointing device.

As an extension to the simple interface described above an interface which allows natural navigation and modification of the environment will be designed. This system will use a web-cam in conjunction with various image processing and computer vision algorithms to allow the user to use their hand as a pointing device. The web-cam will be attached to the users head at approximately eye level and will be pointed back at the screen. From this the system will determine how the user is occluding the screen and in turn which actions they wish to perform.

The vision based scheme will use object tracking, background subtraction and minimal use of hand pose estimation to create a functional input device from the users hand, hand occlusion, pose and position relative to the computer monitor.

2.2 Terrain Deformation

Ideal deformation would allow fine or coarse detail to be added accurately at any chosen position on the terrain. Because an infinitely fine mesh is not a possible representation, the creation of new geometry is an obvious solution to the problem of producing fine detail on a discretised terrain mesh. This on-the-fly creation of new geometry can give rise to unbounded increases in both memory and processing time and, as a result, decreases the frame-rate. For this reason the implementation of two different methods will be considered; the first being the above-mentioned process and the second, a texture-based approach that mimics the desired deformations. These are explained in the following two sub- sections. The development of a height map algorithm that will be used by both implementations will also be designed. This will be used to perform an initialisation of the terrain to a pre-deformed state. The possible algorithms that will be used are the diamond square algorithm [Fournier et al. 1982]. The benefit of this algorithm is that it can generate large data sets quickly. The second algorithm is slower however it produces more visually appealing results. This algorithm, developed by R. Krten, is known as the Fault algorithm[Krten 1994].

2.2.1 Geometry Generation

The first deformation implementation derives from the idea of adaptive geometry creation. When a deformation is made on a coarse mesh, the area in question will be tessellated accordingly so that the desired deformation can be realised accurately. With the help of the modern graphics pipeline, specifically the Geometry Shaders of Shader Model 4.0, such tessellations can be evaluated on graphics hardware. This is desirable for a number of reasons: the CPU processing is not affected, large meshes neednt be sent across the relatively high-latency bus and finally, most tessellations algorithms are reasonably parallelisable. A subdivision surface scheme such as Catmull-Clark [Catmull and Clark 1998] allows for the generation of an approximation to bicubic b-spline patches at each control-mesh face. The use of such tessellations produces a fine local discretisation allowing high-detail displacement maps [Cook 1984] to be realised. The implementation can be performed efficiently using Geometry Shaders [Kazakov 2007]. This real-time adaptive creation of new geometry is inherently unbounded and can cause large demands on graphics resources, however attempts have been made to prevent this [Moule and McCool 2002]. As an attempt to address this problem, a decaying LOD (Level-of-Detail) scheme will be used whereby the detail of any previously created deformations will decrease over time. In addition to this a view-dependent LOD scheme will be employed in order to reduce the amount of invisible geometry that is rendered.

2.2.2 Texture Trickery

The second implementation will be based on texture trickery and employ such techniques as; normal mapping [Gross et al. 2000], parallax mapping [Kaneko et al. 2001] and displacement mapping [Cook 1984] are some of them. These techniques all work on similar principals; those being that they all work using textures.

Bump mapping is a collective term for the normal, parallax and other similar techniques; where by the textures are used to add back higher detail to simplified proxy geometry [Gross et al. 2000] Bump mapping works by using textures in the rendering process to alter the results without modifying the underlying mesh surface. Normal mapping perturbs the surface normals in the lighting calculation to produce the look of a bumpy surface [Wang and Sun 2004].

Displacement mapping on the other hand ”displaces” the mesh's surface, and then these altered vertices are used in the lighting calculations [Hirche et al. 2004]. This technique produces the most accurate results and accounts for various limitations that are present in bump mapping techniques. However full displacement mapping will not be utilised as it incorporates surface tessellation [Hirche et al. 2004] which is not the focus of this implementation. Instead a static mesh that has been pre-tessellated will be used as the surface to apply the deformations on to. A combination of the above two techniques will be used in a level of detail system that will ensure that the program runs in real-time and provide adequate visuals.

2.3 Prototypes

Throughout the development of the program various prototypes will be produced and evaluated along the way. These prototypes will be evaluated after certain milestones have been passed which is to make sure the project remains on track to meet the future deadlines. Each of the three different sections (interface, geometry and textures) will need to be able to stand on their own and be independent of each other. At the end of the project the three sections will be linked together to create the final program. This is to ensure that the sections can be evaluated independently.

The project is going to be designed in C++ and OpenGL which makes it platform independent. OpenGL 3.2 is the graphics library version that will be used as it features the newer hardware functions that will be needed in the implementation process.

The major challenges that would arise during this project would be the actual integration of the three sections in the end. Problems could arise when sections do not fit correctly and cause issues with getting cross section communication.

Because the two implementations are completely different they need to be stand-alone and in order to choose between them different commands need to be executed to launch the different version. This will be done so that code is not redundantly reproduced but rather different methods are invoked to call the desired implementation. The tessellation scheme used in the geometry deformation prototype will be simplified Phong tessellation method [Boubekeur and Alexa 2008].

The interface will be designed to operate in a modular fashion so as to facilitate the use of both methods. These methods will be interchangeable which will allow the techniques to be compared effectively. Furthermore, the control devices (mouse, vision) will be implemented as modules so that they can be easily compared and changed.

2.4 Evaluation

2.4.1 Interface Evaluation

In order to test the functionality of the terrain interface the vision driven interface will be compared against the simpler point and click interface. This evaluation will be done through user testing, whereby users will be asked to accomplish various tasks - such as apply a deformation or move around the interface. The purpose of this evaluation will be to research the efficacy and feasibility of direct manipulation and vision based interfaces.

2.4.2 Evaluation of Deformation Implementations

The deformation techniques will be evaluated using a frame-rate metric. Deformations will be performed using a polyhedral tool called a stamp in order to provide a method of consistent deformation. This stamp will cause an imprint in the terrain in the shape of the stamps leading face. The number of stamp imprints will be used as the independent variable and the resulting frame-rate will be the measured dependent variable. The two techniques will each handle the imprint in their own specific manner and will thus produce different dependence of frame-rate on the number of imprints.

In order to evaluate the quality of mesh deformation an error metric will be used that calculates the difference between the actual deformed surface position and the vertex generated by the algorithm. To do this an offline mesh deformation will be generated on a very fine mesh. Many samples will be taken from each mesh and these sample errors will be combined in a Least-Squares approach to calculate the overall error.

3 Ethics, Professional and Legal Issues

The research question enquires about the possibility of a real-time scheme providing terrain deformation. The results are therefore not based on any user interactions or usability, but will instead be measured by the qualitative and quantitative output and performance of the produced algorithms. However, the evaluation of the computer vision interface requires usability testing and thus produces a need for ethical clearance. In order to conduct user testing, ethical clearance will need to be obtained from the university. All the users will need to be informed of the terms, be told their results are confidential and that they may leave at any time. They will also be asked to sign a form stating that they understand the terms and give us permission to use their results.

There would be no issue relating to intellectual property rights because all of the content is generated through the use of the program. Standard mesh objects will be created to be used as the terrain that gets deformed. This also means that there are no legal issues relating to rights for use of any material in the project.

4 Related Work

Although minimal work has been published regarding real-time deformation of terrains, some techniques for arbitrary mesh deformation are relevant to our implementations. The work of Huang and Wang [Huang et al. 2007] demonstrates the use of local tessellation in order to apply high-detail displacement maps, however their technique is aimed at 3D modelling and cannot be performed in real-time. Similarly, the work of Moule and McCool [Moule and McCool 2002] provides a scheme for adaptive tessellation on the GPU but requires hardware that is not currently in existence.

Much work has been done on tangible and direct manipulation interfaces. While none of these authors use the same approach, their work is relevant to the implementation of the software interface. A system was developed by Piper et al. whereby the users could directly manipulate an environment terrain using their hands [Piper and Ratti 2002]. Their system used mouldable clay onto which the environment height map was projected. A laser scanner detected the height and modifications of the medium. Another system used light pens and a camera projector system to allow direct manipulation[Piazza and Fjeld 2007]. One of the applications was to fly around a terrain using the light pens as natural navigators. The environment was provided and displayed through the Google Earth application.

5 Anticipated Outcomes

5.1 Software created

A software implementation will be completed in order to evaluate the proposed methods and techniques. This system will be capable of displaying terrain data sets and allowing their deformation through a natural interface. This implementation will include novel algorithms for the display of terrain and terrain deformations.

5.2 Expected Impact

Should the program produce good results this system could be used in new computer games. Computer games currently lack terrain that is fully interactive and could greatly benefit from a suitable implementation. There are many other areas that could benefit from the skills gained through the success of this project. The results expected to be gained from this program will prove that current hardware will greatly aid in situations that currently cannot be completed in a real-time environment.

5.3 Key Success Factors

5.3.1 Interface

Success factors for the interface implementation will be measure how easily and rapidly the user can navigate and apply deformations in each respective interface control system.

5.3.2 Reference Geometry Implementation

If this implementation manages to produce a system in which many deformations of the terrain do not lower the frame-rate below acceptable real-time frame-rates (i.e. 30fps). Concurrently, the system should meet requirements imposed on the visual appearance to ensure a level of realism.

5.3.3 Texture Trickery Implementation

The success of this method would be to produce similar results to the reference implementation in terms of visual aesthetics but giving better performance. Another success factor would be the ability for the program to perform well and in real-time given a large number of deformations applied to the terrain.

6 Project Plan

The outline of the risk together with proposed solutions are described below. The timeline together with the noted milestones and requirements are listed as well. This section shows how and when the project will be completed.

6.1 Risks

Here the identifiable risks are discussed and possible solutions to them are given. This includes as many risks that can be thought of in advance. The unforeseen risks will be dealt with as and if they arise.

6.1.1 Access to Hardware

A primary risk would be that of not having access to the required hardware to develop the program. This does not mean access to high-end hardware but at least the minimum spec required for functionality. At present time the university lacks the hardware support and personal equipment will be made use of until such a time that the university can provide the necessary hardware.

6.1.2 Failure to Maintain Real-time Frame-rates

Another risk would be the failure to maintain a real-time system. As mentioned with regard to the geometry shader implementation, adding large amounts of deformations and detail will eventually slow the system down and mean the system fails to maintain real-time. This is a major risk and extensive work will need to be done to try and accommodate for this. Another part that has to be kept real-time would be the computer vision interface which relies on real-time interaction with the computer.

6.1.3 Hardware Limitation

A risk could be that the hardware is not powerful enough to sustain a fully visually appealing real-time system. This would mean the visual quality would need to be reduced to in order to bring the system back into real-time. As hardware improves the quality should improve as well.

6.2 Limitations

The techniques described above are going to be designed for a single computer but if they were to be implemented in a computer game featuring multi-player then the deformations would need to be distributed and consistent across the network which has additional limitations such as speed and latency to deal with. The current design is intended for single computer use and thus this is an acceptable limitation of the design.

6.3 Timeline

The project workload can be divided into a number of sections corresponding to different deliverables and to developer allocations. The Gantt chart in Appendix A describes the composition of these sections and their allocated time-lines.

6.4 Required Resources

The primary resource that this project requires would be the use of hardware based graphical processing units. This hardware is required to visualise the system as well as to perform the underlying calculations. The computer vision system will require a sufficiently capable web-cam to feed data and control the required algorithms.

The only special software requirements are the use of C++ with OpenGL 3.2 which are both freely available. In terms of the people factor, the skills of the authors and the use of test users to evaluate the final product.

6.5 Deliverables

Final Software Implementation	4 October
Report	1 November
Poster	4 November
Web Page	8 November
Reflection Paper	12 November
Presentation	18 November

6.6 Milestones

The milestones below correspond to the completion of certain tasks depicted in the Gantt Diagram. shown in Appendix A.

Initial Design	7 June
LOD Caching	12 July
Point-and-click Interface	21 July
Bump and Displacement Mapping	4 August
Displaced Tessellations	18 August
Final Component Integration	17 September
Report Completed	1 November

6.7 Work Allocation

The project is divided up into three equal weight components that are to be worked on separately and then combined in the end to create the final complete project. There is an initial group work section to which each member will work on together.

6.7.1 Group Work

The creation of the basic framework to which all the separate sections will be done together as a group which will provide each member with sufficient knowledge and understanding of it. This framework will also account for the output rendering of the content to the display. Andrew and Justin will collaboratively work on the level of detail scheme and caching system that will be shared by the two deformation implementations.

6.7.2 Peter Juritz

Implementation of the display and edit interfaces including simple point and click model as well as vision based interface. This interface will display the rendered environment as calculated by both techniques. Functionality to edit the terrain and apply deformations will be included in this. Two control models will be implemented as software modules. The first will be the simple point and click control system. The second will be the vision based interface; this module will implement object tracking, background subtraction and minimal pose estimation. The rendering system which is responsible for displaying the output of the program and the creation of the height-map algorithm that is used to set up the initial scene will be implemented as well.

6.7.3 Andrew Flower

Implementation of the reference technique employing real-time tessellation. This involves code to generate terrain height maps or control meshes for subdivision surfaces, LOD schemes for height map selection and all GLSL shader code required for the real-time generation of tessellated geometry. An input processing component will also be required in order to let the interface component manipulate the geometry.

6.7.4 Justin Crause

Implementation of the texture based approach to terrain deformation. Bump and displacement mapping techniques will be utilised together with a level of detail system to account for visual aesthetics with acceptable real-time performance. Communication with the interface module to retrieve information about new deformations that needs to be applied. Performance data will also need to be calculated to show the performance of the current implementation.

References

BOUBEKEUR, T., AND ALEXA, M. 2008. Phong tessellation. In *SIGGRAPH Asia '08: ACM SIGGRAPH Asia 2008 papers*, ACM, New York, NY, USA, 1–5.

CATMULL, E., AND CLARK, J. 1998. Recursively generated b-spline surfaces on arbitrary topological meshes. 183–188.

COOK, R. L. 1984. Shade trees. In *SIGGRAPH '84: Proceedings of the 11th annual conference on Computer graphics and interactive techniques*, ACM, New York, NY, USA, 223–231.

FOURNIER, A., FUSSELL, D., AND CARPENTER, L. 1982. Computer rendering of stochastic models. *Commun. ACM 25*, 6, 371–384.

GROSS, E. M., HOPGOOD, F. R. A., TARINI, M., CIGNONI, P., ROCCHINI, C., AND SCOPIGNO, R., 2000. Real time, accurate, multi-featured rendering of bump mapped surfaces.

HIRCHE, J., EHLERT, A., GUTHE, S., AND DOGGETT, M. 2004. Hardware accelerated per-pixel displacement mapping. In *In Proceedings of Graphics Interface (2004)*, 153–158.

HOPPE, H. 1996. Progressive meshes. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, ACM, New York, NY, USA, 99–108.

HUANG, X., LI, S., AND WANG, G. 2007. A gpu based interactive modeling approach to designing fine level features. In *GI '07: Proceedings of Graphics Interface 2007*, ACM, New York, NY, USA, 305–311.

KANEKO, T., TAKAHEI, T., INAMI, M., KAWAKAMI, N., YANAGIDA, Y., MAEDA, T., AND TACHI, S. 2001. Detailed shape representation with parallax mapping. In *In Proceedings of the ICAT 2001*, 205–208.

KAZAKOV, M. 2007. Catmull-clark subdivision for geometry shaders. In *AFRIGRAPH '07: Proceedings of the 5th international conference on Computer graphics, virtual reality, visualisation and interaction in Africa*, ACM, New York, NY, USA, 77–84.

KRTEN, R. 1994. Generating realistic terrain. *Dr. Dobbs Journal: Software Tools for the Professional Programmer 19*, 7, 26–28.

LOSASSO, F., AND HOPPE, H. 2004. Geometry clipmaps: terrain rendering using nested regular grids. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers*, ACM, New York, NY, USA, 769–776.

MOULE, K., AND MCCOOL, M. D. 2002. Efficient bounded adaptive tessellation of displacement maps. In *In Graphics Interface*, 171–180.

PATNEY, A., EBEIDA, M. S., AND OWENS, J. D. 2009. Parallel view-dependent tessellation of catmull-clark subdivision surfaces. In *HPG '09: Proceedings of the Conference on High Performance Graphics 2009*, ACM, New York, NY, USA, 99–108.

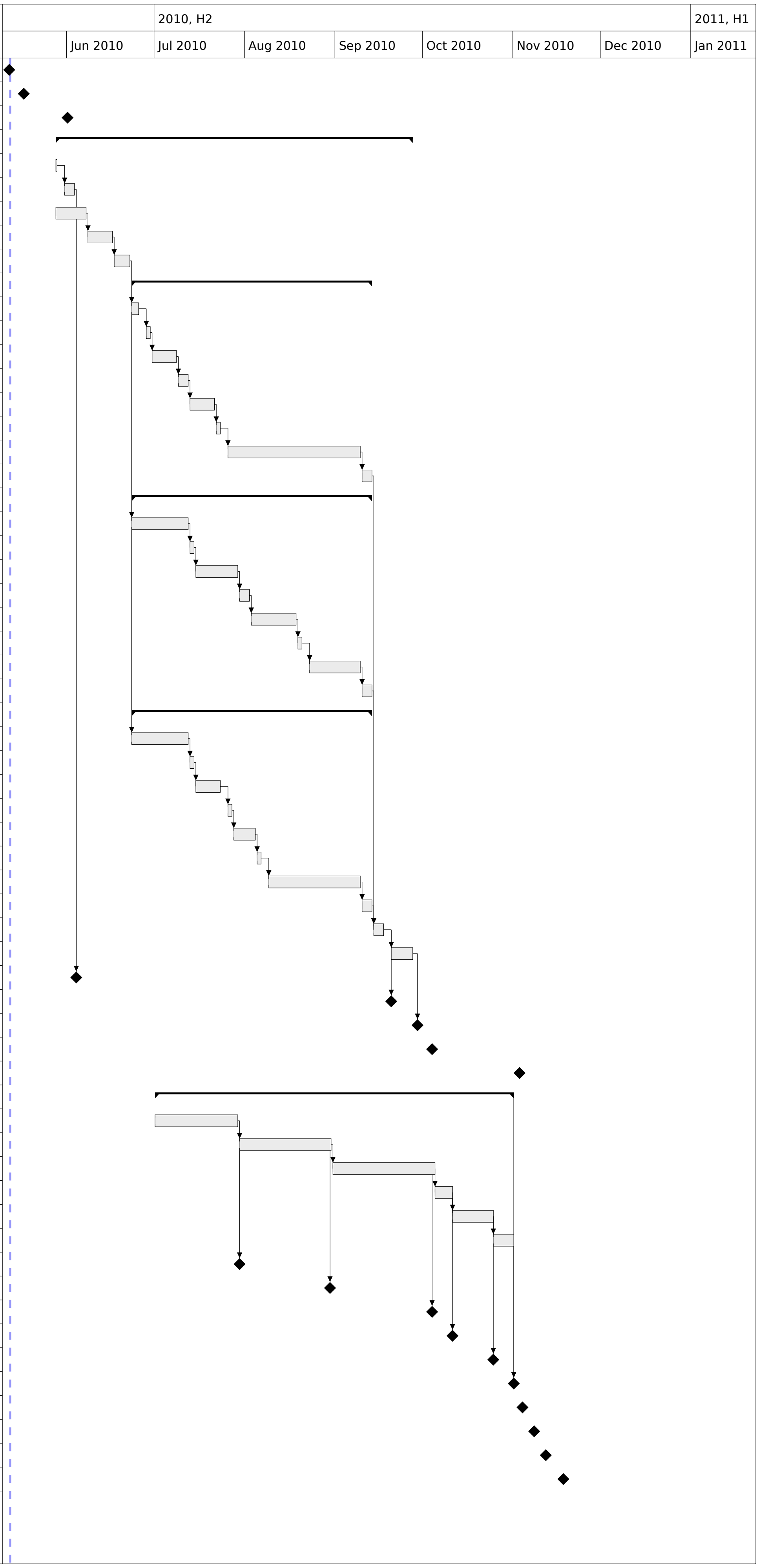
PIAZZA, T., AND FJELD, M. 2007. Ortholumen: Using light for direct tabletop input. *Horizontal Interactive Human-Computer Systems, International Workshop on 0*, 193–196.

PIPER, B., AND RATTI, C. 2002. Illuminating clay: a 3-d tangible interface for landscape analysis. ACM Press, 355–362.

WANG, J., AND SUN, J. 2004. Real-time bump mapped texture shading based-on hardware acceleration. In *VRCAI '04: Proceedings of the 2004 ACM SIGGRAPH international conference on Virtual Reality continuum and its applications in industry*, ACM, New York, NY, USA, 206–209.

Appendix A

Name	Work
Proposal Due	
Proposal Presentation	
Web Site	
Terrain Deform Testbed	210d
Prototype Designs	1d
Prototypes	4d
Design	7d
Codebase, Testbed	7d
Testing and Fixing	4d
User Interface	59d
Heightmap Generation	3d
Testing	2d
Renderer	7d
Testing	2d
Point-and-click	7d
Testing	2d
Computer Vision stuff	34d
Testing	2d
Geometry Deformation	59d
LOD Caching	14d
Testing	2d
Tessellation scheme	11d
Testing and Fixing	2d
Adaptive Tessellation + Displacements	12d
Testing	2d
LOD and manipulation interface	14d
Testing	2d
Texture Deformation	59d
LOD Caching	14d
Testing	2d
Normal, Parallax Mapping	7d
Testing	2d
Displacement Mapping	6d
Testing	2d
Real-time Texture Creation	24d
Testing	2d
Component Integration + Testing	4d
Final Tweaks and Fixes	6d
Feasability Demo	
First Implementation Test	
Final Implementation Test	
Final Implementation Done	
Final Demonstration	
Report	87d
Background	21d
Design	22d
Implementation & Testing	25d
Outline	4d
Draft	10d
Final	5d
Background Chapter Due	
Design Chapter Due	
Implementation & Testing Chapters Due	
Report Outline Due	
Report Draft Due	
Report Final Due	
Poster	
Web Page	
Reflection Paper	
Project Presentation	


2010, H2
2011, H1
Jun 2010
Jul 2010
Aug 2010
Sep 2010
Oct 2010
Nov 2010
Dec 2010
Jan 2011