

Datum	Aktivität	Stunden
29.10.2015	Einrichten der Programmierungsumgebung Xcode. Verstehen von GLFW und Erstellen einiger Beispielprogramme zum Ausprobieren vorhandener Funktionalität. Erstellen von GLSL Vertex- und Fragmentshader zum Darstellen eines einfarbigen Dreiecks.	4
30.10.2015	Implementieren eines ShaderManagers zum komfortablen Laden und Debuggen von Vertex- und Fragmentshadern. Erweitern des vorhandenen Vertex- und Fragmentshader zur Interpolierung der Farben eines Dreiecks.	3
31.10.2015	Recherche über freie Bewegung im dreidimensionalen Raum zur Umsetzung einer Camera Klasse. Implementieren einer Klasse Camera zur freien Bewegung im Raum. Implementieren einer Klasse Keyboard, die es mir erlaubt abzufragen, ob eine Taste gedrückt ist oder nicht. Implementieren einer Funktion mouse_callback die es mir erlaubt Mausbewegungen auf die Kamera umzusetzen.	5
01.11.2015	Implementieren eines eigenen MeshLoaders zum Laden von .obj Dateien. Erfolgreiches Laden eines Würfels.	3
02.11.2015	Implementieren eines Blinn-Phong Shading Models zur schöneren Darstellung.	2
04.11.2015	Meeting mit Prof. Harders	2
05.11.2015	Erweitern des MeshLoaders. Umbauen auf indiziertes Laden von Objekten. Implementierung der Berechnung von Normals.	4
06.11.2015	Unterscheidung zwischen flat-shaded und smooth-shaded Models und der Berechnung von Normals. Implementierung von Vererbung.	3
07.11.2015	Code refactoring und vertraut machen wie man GitHub mit Xcode verwendet. Erster commit.	2
08.11.2015	Neuimplementierung des MeshLoaders, da meine eigene Implementierung nicht mit großen Objekten zurechtkam. Verwendung des Librarys Assimp zum Laden von .obj Dateien. Unterscheidung zwischen Flat- und Smoothshaded Models ist nun nicht mehr notwendig. Als Test wurde ein Stanford Bunny geladen. Es hat einwandfrei funktioniert. Ändern des Vertex- und Fragmentshader damit Konstanten dynamisch über Uniforms geändert werden können. Erstellen einer Klasse Light die es ermöglicht das Licht dynamisch zu bewegen. Erstellen einer Klasse Material die es ermöglicht das Material dynamisch zu ändern.	10
09.11.2015	Implementieren einer Klasse Heightmap die es ermöglicht eine beliebig große flache Fläche, bestehend aus Dreiecken zu erstellen. Die Ecken der Dreiecke sind indiziert und es ist somit auch möglich große Heightmaps effizient zu laden.	4
10.11.2015	Erweitern der Klasse Heightmap um es zu ermöglichen die Höhe beliebiger Vertices zu verändern. Vertrautmachen mit dem Headerfile stb_image.h zum Laden von Bilddateien (.png). Verwenden der schwarz-weiß Bildinformationen zum erfolgreichen Laden einer Heightmap.	3
13.11.2015	Überlegen des Aufbaus der Initialpräsentation. Recherche.	2
14.11.2015	Erweitern des Aufbaus der Initialpräsentation. Recherche.	2
15.11.2015	Erweitern des Aufbaus der Initialpräsentation. Erstellen und Suchen von geeigneten Grafiken.	3
19.11.2015	Umsetzen der Initialpräsentation in Latex	3
20.11.2015	Umsetzen der Initialpräsentation in Latex	2
23.11.2015	Umbau der Präsentation auf das IGS Präsentationstemplate. Umsetzen der vorgeschlagenen Änderungen.	3

Sheet1

25.11.2015	Meeting mit Prof. Harders Finalisieren der Initialpräsentation	2
19.12.2015	Recherche Diamond-Square-Algorithmus	3
20.12.2015	Implementierung des Diamond-Square-Algorithmus Project cleanup (Refactoring)	7
24.12.2015	Recherche Fault-Algorithmus	2
25.12.2015	Implementierung des Fault-Algorithmus	4
04.01.2016	Recherche RMP-Algorithmus	3
05.01.2016	Weitere Recherche RMP-Algorithmus Recherche Graphentheorie Erste Schritte zur Implementierung des RMP-Algorithmus (noch kein Resultat)	5
06.01.2016	Weitere Recherche RMP-Algorithmus Weitere Recherche Graphentheorie Implementierung einer Graphdatenstruktur Implementierung des RMP-Algorithmus samt Zykluserkennung, Ray-Ray-Intersection und IsPointInPolygon Funktion. Resultat des RMP-Algorithmus sichtbar, aber soweit noch nicht zufriedenstellend.	10
07.01.2016	Code refactoring. Weitere Recherche in der Graphentheorie. Graph bugfixes. Neuimplementierung der Zykluserkennung. Finden der minimalen Zyklen im Graphen ist nun verlässlich möglich. RMP bugfixes. Neuimplementierung der Ray-Ray-Intersection. Kleine Änderungen an der IsPointInPolygon Funktion. Erstes zufriedenstellendes Resultat mit RMP.	10
09.02.2016	Recherche geologiebasierte Algorithmen (Erosion, tektonische Plattenbewegungen) Erneute Recherche über tektonische Plattenbewegungen → wenig vielversprechend Recherche Erosion (thermal, hydraulic) → sehr vielversprechend Schreiben des Berichts über geplante geologiebasierte Algorithmen (thermal and hydraulic erosion)	5
10.02.2016	Meeting mit Prof. Harders	1
11.02.2016 - 16.02.2016	Implementieren einer Klasse Cone, die es mir ermöglicht willkürliche Kegel zu erstellen. Diese Kegel werden dann vom Voronoi-Algorithmus verwendet um das Diagramm mithilfe von Z-Buffering auf der GPU zu erstellen. Erstellen einer Klasse TextureManager die es mir ermöglicht Texturen zu laden. Erstellen einer Klasse Texture die alle Informationen über OpenGL Texturen beinhaltet. Erfolgreiches Laden von Texturen mit den GLSL Shadern. Implementieren von höhenabhängiger Texturplatzierung	15

Sheet1

16.02.2016 - 18.02.2016	Implementieren einer Klasse Cube, die es mir ermöglicht willkürliche Würfel zu erstellen. Implementieren einer Klasse Skybox, die alle nötigen Texturen lädt, ein neues ShaderProgram erstellt und dann die Skybox Zeichnet. Refactoring Skybox. Implementieren von Rotationsfunktionalität von Meshes. Implementieren einer Klasse Voronoi, die die Kegel erzeugt, und die per-pixel Informationen und color-checks verwaltet. Implementieren der VoronoiShader die die Kegel mit flat-shading darstellt. Erweitern des RMP Algorithmus damit nun auch die Voronoi-Diagramme verwendet werden.	20
19.02.2016 - 23.02.2016	Erweitern des Fragmentshaders um steigungsbasierte Texturenplatzierung zu ermöglichen. Implementierung von Nebel-effekt in der Ferne. Aufsplitten der Light Klasse in die Unterkategorien PointLight und DirectionalLight. Erweitern der Shader um ein realistischeres Lichtmodell zu ermöglichen. Das Licht kann nun mit Tastatureingabe gesteuert werden. Die Auswirkungen von spekularem Licht sind nun sichtbar.	15
23.02.2016 - 25.02.2016	Recherche über thermale und hydraulische Erosion. Implementierung von thermaler Erosion. Ermöglichen der Echtzeitvisualisierung von thermaler Erosion. Implementieren einer Wasserschicht mit Wellen (sinus und cosinus Wellen). Implementieren eines speziellen Blendings um das Wasser halbdurchsichtig erscheinen zu lassen. Erweitern der Texture Klasse um es zu ermöglichen willkürliche Slots für die Texturen zu wählen. Ändern der Art wie Texturen in OpenGL geladen werden. Implementieren der Texturanimation von Wasser. Fixen eines Bugs im Bezug auf Heightmap Texturekoordinaten um echte Seemless Texturen zu ermöglichen.	20
26.02.2016	Fixen eines ernsten memory leaks.	2
26.02.2016 - 27.02.2016	Säubern des Fragmentshaders von unnötigen Uniforms. Refactoring/Optimierung von main.cpp Übersetzen des Java codes "OpenSimplexNoise" von Kurt Spencer nach C++. Erweitern des Realismus von Wasser mit OpenSimplexNoise. Ändern der Heightmap Klasse, sodass die Vertexkoordinaten nun immer im Bereich zwischen 0 und 1 liegen, egal wie hoch die Auflösung der Heightmap. Einige Änderungen in der Klasse Camera. Die Kamerageschwindigkeit kann nun dynamisch angepasst werden, da die Heightmap Koordinaten nun immer im Bereich zwischen 0 und 1 liegen. Änderungen in anderen Klassen wie RMP, Water aufgrund der Heightmapänderungen. Erweitern thermaler Erosion um Materialeigenschaften zu berücksichtigen. Stein ist hart, Sand ist weich.	15
28.02.2016 - 01.03.2016	Erneute Recherche über thermische und hydraulische Erosion Implementierung hydraulischer Erosion. Neuimplementierung einiger Teile von thermaler Erosion um höheren Realismus zu erreichen. Weitere Änderungen im Bezug auf hydraulische Erosion.	20

Sheet1

02.03.2016	Implementierung von realistischen Wasserreflektionen und -verhalten. Skybox wird nun im Wasser reflektiert. Wassertransparenz ist nun abhängig vom Blickwinkel. Erweitern des ShaderManagers um Validierung zu einem späteren Zeitpunkt zu ermöglichen.	8
03.03.2016 - 04.03.2016	Erweitern der Platzierung von Schnee. Platzierung hängt nun von der Höhe als auch der Steigung des Berges ab. Stein Texturen sind nun auch sichtbar oberhalb der Schneegrenze falls die Steigung hoch genug ist. Skalieren der Texturkoordinaten um mehr Texturdetails darzustellen (4 Tiles teilen dieselbe Textur). Fixen des Fault Algorithmus, dass auch dieser mit dem neuen Heightmap Format (Bereich zwischen 0 und 1) funktioniert. Fixen eines Bugs der immer dann auftrat wenn die Größe des GLFW Fensters verändert wurde.	12
04.03.2016 - 06.03.2016	Fixen eines Bugs des Voronoi Diagrams der dann auftrat wenn die Größe des GLFW Fensters verändert wurde. Implementieren einer Klasse Framebuffer, die es mir erlaubt verschiedene Framebuffer zu verwalten. Implementieren einer Klasse Quad, die es ermöglicht ein einzelnes Quad zu zeichnen. Implementieren eines Bloomfilters, bestehend aus mehreren Shadern/Framebuffers, unter anderem Hochpassfilter, Gaussfilter. Implementieren realistischerer Wassereffekte. Verringerung des Effekts des Bloomfilters auf die Skybox.	15
07.03.2016	Implementierung einer Klasse MousePicker, die es mir ermöglicht Heightmappositionen zu berechnen auf die der Mauszeiger zeigt. Hinzufügen mehrerer Parameter zum RMP Voronoi Algorithmus (Position der Spitze, Ausbreitung des Berges, Zuwachs pro Iteration). Hinzufügen weiterer Checks wenn Heightmap Bilder eingelesen werden. Fixen eines Memory Leaks, das das Erstellen von Voronoi Diagrammen, Kegel, und Meshes generell betraf.	5

07.03.2016 - 09.03.2016	<p>Großflächiges Code Refactoring.</p> <p>Wechsel von Material Pointers zu Material Objects um Speicherverwaltungsprobleme zu vermeiden. Die Mesh Klasse und alle Unterklassen waren von dieser Änderung betroffen.</p> <p>Implementieren einer Klasse Mouse um Informationen über den aktuellen Status der Mausbuttons zwischen zu speichern.</p> <p>Hinzufügen eines Destruktors bei der Klasse Framebuffer, sodass Objekte dieser Art ordentlich gelöscht werden sobald sie nicht mehr gebraucht werden.</p> <p>Setzen der Methode calculateNormals in der Klasse Mesh auf virtual, sodass diese Method von Subklassen überschrieben werden kann.</p> <p>Hinzufügen eines Attributs averageHeight zu Heightmap.</p> <p>Überschreiben der Methode calculateNormals, sodass diese Methode nun auch das Attribut averageHeight updatet.</p> <p>Wechsel von Heightmap pointers auf Heightmap objects um Speicherverwaltungsprobleme zu vermeiden.</p> <p>Mesh Transformationen und Rotationen werden nun mit einer Modelmatrix durchgeführt anstatt die Vertexe direkt zu verändern.</p> <p>Neuimplementierung der Voronoi Klasse.</p> <p>Voronoi generiert nun nicht mehr stratified Kegel. Die Kegel werden nun komplett zufällig platziert.</p> <p>Voronoi verwendet nun eine orthographische Projektion anstatt einer perspektiven.</p> <p>Fixen eines weiteren Bugs bei Voronoi Diagrammen der immer dan auftrat wenn die Fenstergröße verändert wurde.</p> <p>Hinzufügen der Funktionalität zum Erstellen von Screenshots von Voronoi Diagrammen um diesen Algorithmus leichter debuggen zu können.</p> <p>Refactoring des Voronoi codes.</p> <p>Updaten der Voronoishader damit sie mit den Änderungen im Code zusammenspielen.</p>	20
10.03.2016	Recherche über ShallowWater Modelle	5
11.03.2016 - 12.03.2016	<p>Implementierung eines ShallowWater Modells um das realistische Fließen von Wasser auf der Oberfläche des Gebirges zu simulieren. Das Modell ist ein wenig langsam und es kann bei falsch gewählten Parametern zu Oszillationen kommen.</p> <p>Die Maus kann ab sofort frei bewegt werden solange die Ctrl-Taste gedrückt wird. Diese Änderung ermöglicht das präzise Anwenden des RMP Algorithmus.</p>	10
13.03.2016 - 14.03.2016	<p>Fixen des ShallowWater Algorithmus, sodass bei Höhenänderungen des Wassers nicht direkt auf die Heightmap geschrieben wird, sondern zuerst auf eine temporäre Heightmap. Dies ermöglicht es Höhenänderungen der darunterliegenden Landschaft zu berücksichtigen.</p> <p>Die TemporaryHeightmap ist nun nicht mehr nur eine innere Klasse von HydraulicErosion, sondern eine eigenständige Klasse.</p> <p>Das aktuelle ShallowWater Modell ist langsam und unrealistisch. Deshalb wurde ein neues Modell adaptiert von einem physikalisch-gesehen realistischem Modell. Dieses Modell ist recht schnell. Das Modell ist aber nicht im Stande Höhenänderungen der Landschaft zu berücksichtigen und es kann sehr leicht zu Oszillationen kommen.</p> <p>Kleine Änderungen am Fragmentshader um Wasser transparenter erscheinen zu lassen.</p>	12

Sheet1

15.03.2016 - 18.03.2016	<p>Erneute Recherche über ShallowWater-Modelle</p> <p>Neuimplementierung des ShallowWater-Modells nach dem Paper "Fast Hydraulic Erosion Simulation and Visualization on GPU".</p> <p>Die Implementierung wurde auf der CPU durchgeführt. Dieses Modell funktioniert zwar besser als das erste und ist ein wenig schneller, jedoch kann es leicht zu Oszillationen kommen und von Echtzeitausführung ist dieses Modell weit entfernt.</p> <p>Entfernen des physiknahen Modell, da es mir nicht möglich war dieses Modell so anzupassen, dass auch die darunterliegende Landschaft vom Modell berücksichtigt wird.</p>	15
19.03.16 - 21.03.2016	<p>Implementierung eines neuen ShallowWater-Modells nach Trevor Dixon. Dieses Modell ist viel schneller als alle vorherigen Modelle und kann mühelos auf der CPU ausgeführt werden.</p> <p>Regentropfen fallen gleichmäßig verteilt auf der ganzen Heightmap.</p> <p>Tropfenattribute wie z.B. Gewicht oder Radius können durch Parameter gesetzt werden.</p> <p>Im Moment wird vom ShallowWater-Modell die darunterliegende Landschaft noch nicht berücksichtigt.</p> <p>Einlesen in die ffmpeg Bibliothek.</p> <p>Implementierung einer Klasse Util, die es mir ermöglicht Screenshots (.png) und Videos (.mpg MPEG2 codec) zu erstellen.</p> <p>Ersetzen der Screenshotfunktionalität der Voronoi Klasse mit der von Util.</p>	15