| Datum | Aktivität | Stunden |
|---|---|---|
| 29.10.2015 | Einrichten der Programmierumgebung Xcode.<br>Verstehen von GLFW und Erstellen einiger Beispielprogramme zum Ausprobieren vorhandener Funktionalität.<br>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.<br>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.<br>Implementieren einer Klasse Camera zur freien Bewegung im Raum.<br>Implementieren einer Klasse Keyboard, die es mir erlaubt abzufragen, ob eine Taste gedrückt ist oder nicht.<br>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.<br>Erfolgreiches Laden eines Würflels. | 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.<br>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.<br>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.<br>Ändern des Vertex- und Fragmentshader damit Konstanten dynamisch über Uniforms geändert werden können.<br>Erstellen einer Klasse Light die es ermöglicht das Licht dynamisch zu bewegen.<br>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.<br>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.<br>Vertrautmachen mit dem Headerfile stb_image.h zum Laden von Bilddateien (.png).<br>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 geigneten 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 |

| | | |
|---|---|---|
| 25.11.2015 | Meeting mit Prof. Harders<br>Finalisieren der Initialpräsentation | 2 |
| 19.12.2015 | Recherche Diamond-Square-Algorithmus | 3 |
| 20.12.2015 | Implementierung des Diamond-Square-Algorithmus<br>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<br>Recherche Graphentheorie<br>Erste Schritte zur Implementierung des RMP-Algorithmus (noch kein Resultat) | 5 |
| 06.01.2016 | Weitere Recherche RMP-Algorithmus<br>Weitere Recherche Graphentheorie<br>Implementierung einer Graphdatenstruktur<br>Implementierung des RMP-Algorithmus samt Zykluserkennung, Ray-Ray-Intersection und IsPointInPolygon Funktion.<br>Resultat des RMP-Algorithmus sichtbar, aber soweit noch nicht zufriedenstellend. | 10 |
| 07.01.2016 | Code refactoring.<br>Weitere Recherche in der Graphentheorie.<br>Graph bugfixes. Neuimplementierung der Zykluserkennung. Finden der minimalen Zyklen im Graphen ist nun verlässlich möglich.<br>RMP bugfixes. Neuimplementierung der Ray-Ray-Intersection. Kleine Änderungen an der IsPointInPolygon Funktion.<br>Erstes zufriedenstellendes Resultat mit RMP. | 10 |
| 09.02.2016 | Recherche geologiebasierte Algorithmen (Erosion, tektonische Plattenbewegungen)<br>Erneute Recherche über tektonische Plattenbewegungen → wenig vielversprechend<br>Recherche Erosion (thermal, hydraulic) → sehr vielversprechend<br>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.<br>Erstellen einer Klasse TextureManager die es mir ermöglicht Texturen zu laden.<br>Erstellen einer Klasse Texture die alle Informationen über OpenGL Texturen beinhaltet.<br>Erfolgreiches Laden von Texturen mit den GLSL Shadern.<br>Implementieren von höhenabhängiger Texturplatzierung | 15 |

| | | |
|---|---|---|
| 16.02.2016 - 18.02.2016 | Implementieren einer Klasse Cube, die es mir ermöglicht willkürliche Würfel zu erstellen.<br>Implementieren einer Klasse Skybox, die alle nötigen Texturen lädt, ein neues ShaderProgram erstellt und dann die Skybox Zeichnet.<br>Refactoring Skybox.<br>Implementieren von Rotationsfunktionalität von Meshes.<br>Implementieren einer Klasse Voronoi, die die Kegel erzeugt, und die per-pixel Informationen und color-checks verwaltet.<br>Implementieren der VoronoiShader die die Kegel mit flat-shading darstellt.<br>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.<br>Implementierung von Nebeleffekt in der Ferne.<br>Aufsplitten der Light Klasse in die Unterkategorien PointLight und DirectionalLight.<br>Erweitern der Shader um ein realistischeres Lichtmodell zu ermöglichen.<br>Das Licht kann nun mit Tastatureingabe gesteuert werden.<br>Die Auswirkungen von spekularem Licht sind nun sichtbar. | 15 |
| 23.02.2016 - 25.02.2016 | Recherche über thermale und hydraulische Erosion.<br>Implementierung von thermaler Erosion.<br>Ermöglichen der Echtzeitvisualisierung von thermaler Erosion.<br>Implementieren einer Wasserschicht mit Wellen (sinus und cosinus Wellen).<br>Implementieren eines speziellen Blendings um das Wasser halbdurchsichtig erscheinen zu lassen.<br>Erweitern der Texture Klasse um es zu ermöglichen willkürliche Slots für die Texturen zu wählen.<br>Ändern der Art wie Texturen in OpenGL geladen werden.<br>Implementieren der Texturanimation von Wasser.<br>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.<br>Refactoring/Optimierung von main.cpp<br>Übersetzen des Java codes “OpenSimplexNoise” von Kurt Spencer nach C++.<br>Erweitern des Realismus von Wasser mit OpenSimplexNoise.<br>Ändern der Heightmap Klasse, sodass die Vertexkoordinaten nun immer im Bereich zwischen 0 und 1 liegen, egal wie hoch die Auflösung der Heightmap.<br>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.<br>Änderungen in anderen Klassen wie RMP, Water aufgrund der Heightmapänderungen.<br>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<br>Implementierung hydraulischer Erosion.<br>Neuimplementierung einiger Teile von thermaler Erosion um höheren Realismus zu erreichen.<br>Weitere Änderungen im Bezug auf hydraulische Erosion. | 20 |

| | | |
|---|---|---|
| 02.03.2016 | Implementierung von realistischen Wasserreflektionen und -verhalten.<br>Skybox wird nun im Wasser reflektiert.<br>Wassertransparenz ist nun abhängig vom Blickwinkel.<br>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.<br>Stein Texturen sind nun auch sichtbar oberhalb der Schneegrenze falls die Steigung hoch genug ist.<br>Skalieren der Texturkoordinaten um mehr Texturdetails darzustellen (4 Tiles teilen dieselbe Textur).<br>Fixen des Fault Algorithmus, dass auch dieser mit dem neuen Heightmap Format (Bereich zwischen 0 und 1) funktioniert.<br>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.<br>Implementieren einer Klasse Framebuffer, die es mir erlaubt verschiedene Framebuffer zu verwalten.<br>Implementieren einer Klasse Quad, die es ermöglicht ein einzelnes Quad zu zeichen.<br>Implementieren eines Bloomfilters, bestehend aus mehreren Shadern/Framebuffern, unter anderem Hochpassfilter, Gaussfilter.<br>Implementieren realistischerer Wassereffekte.<br>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.<br>Hinzufügen mehrerer Parameter zum RMP Voronoi Algorithmus (Position der Spitze, Ausbreitung des Berges, Zuwachs pro Iteration).<br>Hinzufügen weiterer Checks wenn Heightmap Bilder eingelesen werden.<br>Fixen eines Memory Leaks, das das Erstellen von Voronoi Diagrammen, Kegel, und Meshes generell betraf. | 5 |

| | | |
|---|---|---|
| 07.03.2016 - 09.03.2016 | ~~Großflächiges Code Refactoring.~~<br>Wechsel von Material Pointers zu Material Objects um Speicherverwaltungsprobleme zu vermeiden. Die Mesh Klasse und alle Unterklassen waren von dieser Änderung betroffen.<br>Implementieren einer Klasse Mouse um Informationen über den aktuellen Status der Mausbuttons zwischen zu speichern.<br>Hinzufügen eines Destruktors bei der Klasse Framebuffer, sodass Objekte dieser Art ordentlich gelöscht werden sobald sie nicht mehr gebraucht werden.<br>Setzen der Methode calculateNormals in der Klasse Mesh auf virtual, sodass diese Method von Subklassen überschrieben werden kann.<br>Hinzufügen eines Attributs averageHeight zu Heightmap.<br>Überschreiben der Methode calculateNormals, sodass diese Methode nun auch das Attribut averageHeight updatet.<br>Wechsel von Heightmap pointers auf Heightmap objects um Speicherverwaltungsprobleme zu vermeiden.<br>Mesh Transformationen und Rotationen werden nun mit einer Modelmatrix durchgeführt anstatt die Vertexe direkt zu verändern.<br>Neuimplementierung der Voronoi Klasse.<br>Voronoi generiert nun nicht mehr stratified Kegel. Die Kegel werden nun komplett zufällig platziert.<br>Voronoi verwendet nun eine orthographische Projektion anstatt einer perspektiven.<br>Fixen eines weiteren Bugs bei Voronoi Diagrammen der immer dan auftrat wenn die Fenstergröße verändert wurde.<br>Hinzufügen der Funktionalität zum Erstellen von Screenshots von Voronoi Diagrammen um diesen Algorithmus leichter debuggen zu können.<br>Refactoring des Voronoi codes.<br>Updaten der Voronoishader damit sie mit den Änderungen im Code zusammenspielen. | 20 |
| 10.03.2016 | Recherche über ShallowWater Modelle | 5 |
| 11.03.2016 - 12.03.2016 | 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.<br>Die Maus kann ab sofort frei bewegt werden solange die Ctrl-Taste gedrückt wird. Diese Änderung ermöglicht das prezise Anwenden des RMP Algorithmus. | 10 |
| 13.03.2016 - 14.03.2016 | 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.<br>Die TemporaryHeightmap ist nun nicht mehr nur eine innere Klasse von HydraulicErosion, sondern eine eigenständige Klasse.<br>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.<br>Kleine Änderungen am Fragmentshader um Wasser transparenter erscheinen zu lassen. | 12 |

| | | |
|---|---|---|
| 15.03.2016 - 18.03.2016 | Erneute Recherche über ShallowWater-Modelle<br>Neuimplementierung des ShallowWater-Modells nach dem Paper “Fast Hydraulic Erosion Simulation and Visualization on GPU”.<br>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.<br>Entfernen des physiknahem Modell, da es mir nicht möglich war dieses Modell so anzupassen, dass auch die darunterliegende Landschaft vom Modell berücksichtigt wird. | 15 |
| 19.03.2016 - 21.03.2016 | 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.<br>Regentropfen fallen gleichmäßig verteilt auf der ganzen Heightmap.<br>Tropfenattribute wie z.B. Gewicht oder Radius können durch Parameter gesetzt werden.<br>Im Moment wird vom ShallowWater-Modell die darunterliegende Landschaft noch nicht berücksichtigt.<br>Einlesen in die ffmpeg Bibliothek.<br>Implementierung einer Klasse Util, die es mir ermöglicht Screenshots (.png) und Videos (.mpg MPEG2 codec) zu erstellen.<br>Ersetzen der Screenshotfunktionalität der Voronoi Klasse mit der von Util. | 15 |
| 22.03.2016 - 23.03.2016 | Fixen eines Bugs bei der Videoaufnahme. Es wurde irrtümlich versucht einen nicht belegten Speicherplatz zu befreien.<br>Updaten des Zeitlogs.<br>Recherche über Smoothed Particle Hydrodynamics (SPH).<br>Testen diverser SPH libraries und code repositories. | 12 |
| 24.03.2016 - 25.03.2016 | Suchen und Finden einer potentiellen SPH-Implementation (von Saeed Mahani).<br>Umschreiben einiger Teile der SPH-Implementation.<br>Die originale Implementation verwendet GLUT, mein Projekt verwendet GLFW.<br>GLFW stellt keine Funktionen bereit um Kugeln zu generieren. Diese Funktionalität musste selbst in Form von Icosphären implementiert werden. Die Klasse Icosphere ermöglicht es mir prozedural triangulierte Kugeln beliebiger Genauigkeit zu generieren.<br>Die originale Implementation war nur in der Lage Kugeln in einer Box kollidieren zu lassen jedoch nicht mit beliebiger Geometrie, wie z.B. einer Heightmap. Diese Funktionalität muss dann später selbst dazuimplementiert werden.<br>Um das Rendern der SPH-Simulation zu beschleunigen wurde ein neuer leichtgewichtiger Shader programmiert, der es ermöglicht selbst eine große Anzahl an Partikeln zu rendern.<br>Optimieren des Voronoi Algorithmus sodass nur einmalig eine Kegel Geometrie generiert werden muss, die dann immer wieder wiederverwendet wird.<br>Temporäres deaktivieren des künstlichen Wassers um die Partikelsimulation zu beschleunigen. | 20 |
| 26.03.2016 | Implementieren sogenannter Solid-Particles. Diese “festen” Partikel bewegen sich nicht, jedoch können andere Partikel mit ihnen kollidieren. Diese Partikel werden überall auf der Landschaft verteilt. | 5 |

| | | |
|---|---|---|
| 27.03.2016 - 30.03.2016 | Solid-Particles waren sehr ressourcenhungrig und nicht allzu realistisch. Es wurde eine alternative Kollisionserkennung implementiert mit der die Landschaft direkt in die SPH-Partikelsimulation integriert werden kann.<br>Angenommen ein Partikel befindet sich in der OpenGL x,z-Ebene: Es wird untersucht ob die y-Koordinate des Partikels bereits unter der Landschaft ist oder nicht (Triangle-Ray-Intersection-Test). Falls ja, wird mithilfe des entsprechenden Normalvektors an der x,z-Position der Landschaft die Richtung bestimmt, in welche eine Kraft auf die mit der Landschaft kollidierenden Kugel wirken muss. Erweitern der SPH-Simulation um eine Regen-Funktion die zufällig im Himmel verteilt Regentropfen erscheinen Lässt.<br>Recherche Marching-Cubes Algorithmus.<br>Erweitern der SPH-Simulation um eine Marching-Cubes Implementation. Marching Cubes ist rechentechnisch sehr aufwendig Und kann deshalb auf Knopfdruck ausgeführt werden.<br>Implementieren einer Klasse MetaMesh, die die willkürliche Geometrie des durch den Marching-Cubes Algorithmus generierten Objekts verwaltet. | 25 |
| 31.03.2016 - 01.04.2016 | Implementieren einer Funktion zur Bestimmung der genauen Höhe an einer willkürlichen Position der Heightmap (auch zwischen Gridpunkten). Die Implementation verwendet Möller-Trumbore Ray-Triangle-Intersection.<br>Implementieren einer Funktion zum komfortablen Inkrementieren/Dekrementieren der Höhe an einer beliebigen Position der Heightmap.<br>Erweitern der Klasse Keyboard um eine Zeitstempel-Funktionalität. Bei jedem Tastendruck wird gespeichert, wann die Taste gedrückt wurde. Somit ist es möglich sicherzustellen, dass z.B. boolean Variablen nur einmal getoggelt werden.<br>Erweitern der SPH-Simulation, sodass Partikel Einfluss auf die Landschaft nehmen (hydraulische Erosion).<br>Jeder Partikel hat nun eine Lebensdauer und Sedimentkapazität. Mithilfe der Konstante PARTICLE_ACIDITY ist es möglich festzulegen wie “aggressiv” ein Partikel die Landschaft auflösen soll. | 12 |
| 09.04.2016 | Recherche über tektonische Plattenbewegungen.<br>Schauen eines Dokumentationsfilmes über die Entstehung der Alpen. | 3 |
| 10.04.2016 - 16.04.2016 | Suche nach bereits existierenden Softwarelösungen über tektonische Plattenbewegungen → Nichts vielversprechendes.<br>Überlegen wie man ähnlich wie im Paper “Physically Based Terrain Generation: Procedural Heightmap Generation Using Plate Tectonics” von Lauri Viitanen, die tektonischen Plattenbewegungen umsetzen könnte.<br>Finden des 2D-Physik-Librarys Box2D.<br>Einlesen in die Box2D API.<br>Vertrautmachen mit den Codebeispielen von Box2D. | 12 |
| 17.04.2016 - 22.04.2016 | Verwenden der Box2D Testumgebung “Testbench”. Programmieren eines “Tests” für die Testumgebung von Box2D.<br>Der erste Test beinhaltet einige dynamische Objekte die in einer Zero-Gravity-Umgebung herumschweben und kinematischen Wänden, die sich stets bewegen und mit den dynamischen Objekten kollidieren. So wird sichergestellt, dass die Simulation nie still steht.<br>Erweitern des Tests. Die “Platten” können nun mithilfe der Maus beliebig oft geteilt werden. Es enstehen Polygone mit bis zu 8 Ecken. Diese dynamischen Objekte können wiederum mit anderen dynamischen oder kinematischen Objekten kollidieren. | 12 |

| | | |
|---|---|---|
| 23.04.2016 - 24.04.2016 | Erweitern des Tests. Die tektonische Plattensimulation nimmt nun langsam Form an. Die Simulation beginnt nun mit einer großen Platte, die durch User-Interaktion mithilfe der Maus beliebig oft geteilt werden kann. Die Kollisionserkennung gibt nun in der Konsole alle Koordinaten aus bei denen 2 Polygone kollidiert sind.<br>Das Splitten von Polygonen ist nun viel realistischer. Beim Splitten wird nun die Bewegungs-, als auch die Rotationsgeschwindigkeit der Vaterplatte an die Kindplatten weitergegeben. Aufgrund dieser Energieerhaltungsmaßnahme ist die Simulation viel realistischer. | 8 |
| 25.04.2016 - 04.05.2016 | Filtern von Kollisionen. Kollisionen der dynamischen Objekte mit den kinematischen sollen nicht berücksichtigt werden. Außerdem muss aufgrund der Vielzahl an Kollisionen pro Simulationsschritt vereinfacht werden. Deshalb wurde vorausschauend bereits überlegt wie man am Geschicktesten vorgehen könnte um eine Erhöhung der Landschaft an der Kante zweier kollidierender Platten zu ermöglichen. Schlussendlich wurde folgende Lösung gewählt: Es werden nur Kollisionen Berücksichtigt, bei denen 2 Platten so kollidieren, sodass es genau 2 Kollisionspunkte gibt (perfekte Kollision). Zwischen diesen zwei Punkten kann dann später wenn die Simulation in das Hauptprogramm integriert ist eine Erhöhung der Landschaft hervorgerufen werden.<br>Code Refactoring des Box2D Tests.<br>Platzieren von zufälligen “Schnitten” am Anfang der Simulation. (Der User muss nun nicht mehr selbst die Platten splitten). | 8 |
| 05.05.2016 | Mergen des Box2D Testprogramms mit meinem TerrainGeneration Hauptprogramm.<br>Erstellen einer Klasse TectonicPlateSimulation.<br>Verwenden des RMP Algorithmus um entlang der Kollisionslinie eine prozedurale Gebirgskette zu generieren.<br>Aufgrund der Komplexität des Algorithmus kann die Tektonische Plattensimulation nur schrittweise mittels Tastendruck ausgeführt Werden. | 10 |
| 06.05.2016 - 08.05.2016 | Geschwindigkeitsoptimierungen für das Übertragen der Kollisionsdaten der Tektonischen Plattensimulation auf die Landschaft.<br>Das Projezieren der Simulationsschritte auf die Heightmap benötigt nun weniger als eine Sekunde.<br>Um diesen Geschwindigkeitszuwachs zu ermöglichen war es nötig den Voronoi und RMP Algorithmus zu erweitern.<br>Die Klasse Voronoi wurde um die Funktionalität erweitert, Farben einer beliebigen x,y bzw. column,row Koordinate auszugeben.<br>Die Klasse RMP wurde erweitert um das effiziente Generieren von Gebirgsketten zu ermöglichen.<br>Dazu wurde der Bresenham Linien Algorithmus verwendet, um jene Punkte der Heightmap zu bestimmen, die von einer Kollision zweier Platten betroffen sein könnten.<br>Dadurch muss nun nur mehr ein einziges Voronoi Bild pro RMP Iteration erstellt werden um eine Gebirgskette zu generieren.<br>Die Klasse TectonicPlateSimulation wurde entsprechend angepasst, sodass nun die neuen Implementationen des Voronoi als auch RMP Algorithmus verwendet werden.<br>Hinzufügen der kompletten Box2D Simulation inklusive Testbench in das Backup Verzeichnis. | 10 |
| 11.05.2016 | Erstellen einer GIF-Animation zur Veranschaulichung der tektonischen Plattensimulation im Hauptprogramm.<br>Updaten des Zeitlogs. | 3 |