Semesterarbeit
Fach: Game Engine Architecture WS12/13
Bearbeiter: Steffen Buder

Lunar Lander 3D - Dokumentation

Der Inhalt dieser Dokumentation dient der Beschreibung der im Rahmen der Veranstaltung „Game Engine Architecture“ fertiggestellten Semesterarbeit.
Darin enthalten sind die Beschreibung des Spiels und der implementierten Logik, sowie Probleme und Verbesserungsvorschläge.

Die Implementierungsdetails werden hier nur grob erwähnt und können mit dem Quellcode (https://github.com/BuDDi/GEA) vervollständigt werden.

Änderungshistorie

Vorraussetzungen und Einschränkungen

Für das Spiel wird eine Wii-Fernbedienung benötigt.
Zusätzlich wird das Zusatzprogramm Osculator benötigt bzw. Vergleichbares für andere Systeme.
Osculator bietet die Möglichkeit Nachrichten von Perepheriegeräten zu generieren und diese an Adressen zu verschicken um sie für andere Anwendungen verfügbar zu machen.
Die Implementierung erfolgt dann in Unity ohne nativen Code mittels Oscumote, eine vorgefertigte Scriptsammlung zum Empfangen der Nachrichten, die von Osculator gesendet werden.
Diese Variante mit der Wii-Fernbedienung zu kommunizieren ist die favorisierten Lösung, da diese für die Indie-Version von Unity funktioniert. Die Alternative nativen Code einzubinden kann nur in der Pro-Version von Unity realisiert werden.
Die Tatsache, dass Serverkommunikation stattfindet, hat allerdings zur Folge, dass das resultierende Spiel nicht ohne Weiteres im Webplayer lauffähig ist. Eine Lösung konnte bis zum Ende dieser Semesterarbeit hauptsächlich aus Zeitmangel nicht gefunden werden, was nicht heisst, dass es keine Lösung gibt.

Spielbeschreibung

Das Spiel ist eine Neuauflage des Arcarde-Klassikers Lunar Lander aus dem Jahr 1979. Dabei wurde der damalige Klassiker in die dritte Dimension überführt.

Dem Spieler stehen 2 Levels (Mars & Mond) zur Verfügung.
Dabei bewegt er eine Raumfähre über eine dieser Karten und muss es innerhalb einer vorgeschriebenen Zeit schaffen alle Landeplattformen anzufliegen und dabei möglichst wenig Treibstoff verbrauchen. Ein Pfeil, sowie die kleine Minimap unterstützen den Spieler dabei die Plattformen zu finden. Jede erfolgreich angeflogene Plattform (erfolgreich bedeutet mit allen „4 Füßen“ auf dem Boden) bringt dem Spieler Punkte, die seinem Konto gutgeschrieben werden.
Sind alle Plattformen innerhalb der Zeit angeflogen hat der Spieler die Mission erfolgreich beendet.

Ziel des Spiels ist es für die einzelnen Levels eine möglichst hohe Punktzahl zu erreichen. Das kann erreicht werden indem ein Level innerhalb einer vorgeschriebenen Zeit möglichst schnell und ohne viel Treibstoffverlust beendet wird.

Das Spiel ist beendet wenn ... :
- der Spieler das Level innerhalb der vorgeschriebenen Zeit beendet. Das heisst es wurde auf allen Plattformen gelandet.
- der Spieler die nicht sichtbare Grenze der Karte überschreitet und nicht innerhalb der vorgeschriebenen Zeit wieder in Richtung Pfeil zur Landeplattformzone zurückkehrt.
- die Zeit abgelaufen ist.
- der Treibstoffvorrat aufgebraucht ist.
- die Raumfähre zerstört wurde in Folge von harten Aufprallen während des Spiels.

Hat der Spieler das Level erfolgreich beendet, werden die Punkte in die Punkteliste für das entsprechende Level eingetragen und abgespeichert. Er hat dann die Möglichkeit das Level neu zu starten oder zum Hauptmenü zurückzukehren.
Das Gleiche Menü erscheint auch wenn der Spieler das Ziel nicht erreicht hat.

Bepunktung

Die Punkte für eine erfolgreich angeflogene Plattform setzen sich aus der Grundpunktzahl der Plattform abzüglich des Abstands zur Mitte der Plattform und dem gewonnen Bonusfaktor zusammen. Der Bonusfaktor wird dann bestimmt durch den Faktor der Plattform und der Anzahl der Füße, die bei Erstkontakt die Platform berührt haben.

Die Endpunktzahl wird mit der Levelpunktzahl, dem Treibstoffvorrat und der noch verbleibenden Zeit verrechnet.

Steuerung

Gesteuert wird mit der Wiimote und dem Zusatzcontroller Nunchuk.

Die WiiMote dient der Hauptinteraktion. Das Spiel verlangt zwar den Nunchuk-Controller, aber es kann prinzipiell auch ohne diesen gesteuert werden.

Innerhalb eines Menüs sind die Steuerkreuztasten für die Navigation zuständig. Der Spieler kann dann den entsprechenden Punkt wählen und mit der A-Tasten bestätigen.

Während des Spiels steuert der Spieler mit WiiMote und Nunchuk die Raumfähre.
Mit der Fernbedienung wird die Hauptdüse aktiviert und die Fähre bekommt Schub, abhängig von der Stellung der Hauptdüse.
Die Stellung der Hauptdüse wird beeinflusst von der Stellung der Fernbedienung.
Das heisst, wenn der A-Knopf nach oben zeigt wird dies als Normalstellung empfunden und die Düse zeigt gerade nach unten. Kippt man die Fernbedienung in eine Richtung geht der Schub auch in diese Richtung und die Fähre bewegt sich in diese.

Mit den Links-, Rechts-Tasten lässt sich die Kamera umschalten.
Mit dem Home-Knopf gelangt der Spieler in das Pause-Menü und mit dem A-Knopf können Aktionen innerhalb des Menüs bestätigt werden.

Der Nunchuk-Controller dient der Aktivierung der kleinen Düsen.
Davon besitzt die Raumfähre jeweils eine an den vier Seiten und eine weitere oben drauf, gegenüber der Hauptdüse.
Damit lassen sich komplexe Manöver steuern und der Spieler besitzt volle Rotationsfreiheit auf den xz-Achsen.

Tipp: Die obere Düse eignet sich sehr gut zur Korrektur von misslungenen Manövern und zum schnelleren Landeanflug.

Die obere Düse wird mit der c-Taste aktiviert.
Die seitlichen Düsen werden mit der Z-Taste aktiviert, wenn der Steuerknüppel des Nunchuks zusätzlich in die Richtung gedrückt wird, in die die Raumfähre rotieren soll.

Levels
Zur Auswahl stehen die beiden Levels Mars und Mond.

Die Gravitation der Levels orientiert sich an der der realen Gegenstücke.
Somit lässt sich das Mond-Level mit gleichen Beschleunigungs- und Geschwindigkeitseigenschaften deutlich schwerer steuern und ist das „spätere“ Level.

Implementierung

Der folgende Abschnitt soll grob die Implementierung mit den Kernkomponenten vorstellen. Nähere Details können dem Quellcode entnommen werden (https://github.com/BuDDi/GEA).

Die Implementierung gliedert sich in die folgenden Hauptkomponenten.

Lander (Player)

Der Lander ist im Spiel repräsentiert durch das eigentliche Mesh der Raumfähre und den Partikelsystemen für die einzelnen Düsen, sowie dem Staubemitter.
Diesem Mesh hängen auch diverse Collider an. Jeweils einer für die einzelnen „Füße“ und ein Collider für den Rest des Meshes.

Dieses Prefab (im Projekt LunarLanderBeta) regelt im Prinzip die Steuerung der Raumfähre mit Hilfe der Scripte (LunarLanderWiiMote, Lander).

LunarLanderWiiMote regelt den Input durch die WiiMote, also auch alle Düsen und somit die eigentliche Steuerung des Playerobjekts.
Lander sorgt für die zusätzlichen Attribute der Raumfähre, d.h. Schaden, Treibstoffvorrat. Sofern benötigt bieten diese Komponenten getter- und setter-Methoden zur Verfügung.

Hinweis: Für die drei Kernkomponenten gilt, das nur eins im laufenden Spiel vorhanden ist. Für die Implementierung bedeuten dies, dass das Singleton-Pattern angewendet wurde und das Hauptskript eine Instanz hält. Alle Aktion werden über diese Instanz geregelt. Handelt es sich um ein Skript, welches sich als Komponente an einem GameObject befindet (erbt von MonoBehaviour), wird das Instanzattribut bei Start neu gesetzt. Somit ist gewährleistet, dass die GetInstance-Methode auch das aktuell aktive, im Spiel befindliche Objekt zurückgibt.

LandingPadGroup (LevelManager)

Dieses Script wird zusammen mit dem entsprechenden Prefab verwendet. Und sorgt für die Levellogik. Es hält den aktuellen Punktestand, die Zeit und steuert das Spielgeschehen (Pause, Spielende, auf Plattform landen und Punkte berrechnen etc.).

An dem gleichnamigen Prefab befindet sich auch die Script-Komponente Levelbounds welche die Events des umspannenden Trigger-Colliders verarbeitet und somit die „Rückkehrzeit“ starten. Der Trigger-Collider sollte also immer die Plattformen umspannen, andererseits startet die Zeit auch bei Anflug außerhalb befindlicher Plattformen.

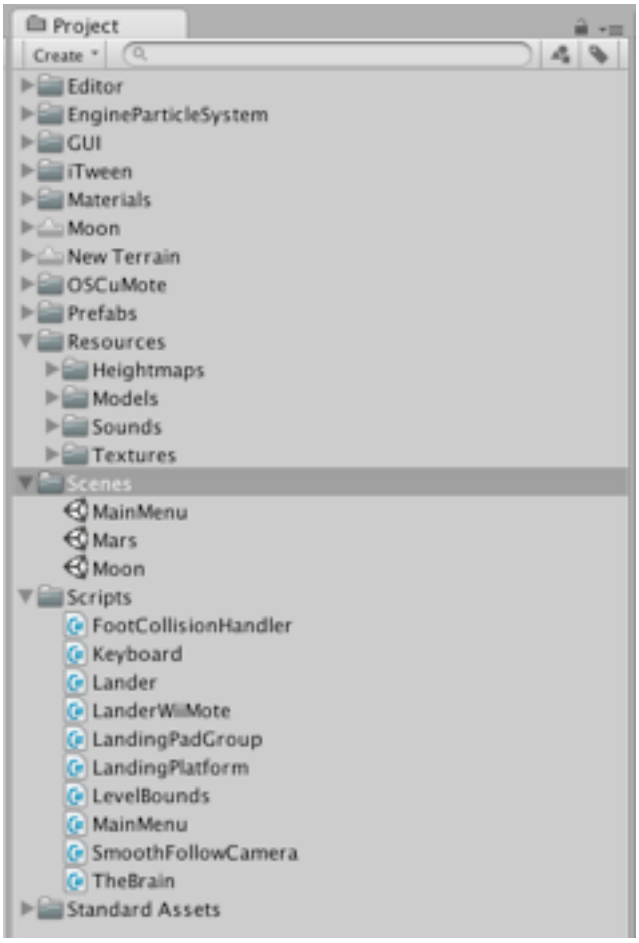
Per Script können andere Komponenten auf die Hauptaktionen Aktivierung von Pause, Auslösen des Spielendes und Darstellung eines Textes in der Bildmitte (MessageLabel) zurückgreifen.

TheBrain (Persistenz)

Diese Scriptkomponente „lebt“ über den gesamten Zeitraum des Spiels und hält die Punkteliste in einem Dictionary mit den Namen der Karten und den dazugehörigen Punktelisten. Sie bietet Methoden zum Laden eines Speicherstandes aus einer Datei und der Auslieferung der eingelesene Daten.
Desweiteren befindet sich hier die Logik für das Einfügen und Abspeichern neuer Punktestände.

Projektstruktur

Das Projekt gliedert sich relativ selbsterklärend in gewisse Hauptordner.
Dennoch soll hier eine Übersicht über die Projektorganisation erfolgen (Text bezieht sich auf das Bild von oben nach unten).



Editor - Standardverzeichnis für Editor-Erweiterungen

EngineParticleSystem - Prefab-Sammlung für Düsenpartikelsysteme; wurde nicht verwendet, da die Partikelsysteme auf dem alten Untiy-Partikelsystem basieren

GUI - enthält alle GUI relevanten Elemente, wie GUISkin, Schriftarten, GUI-Texturen

iTween - freie Erweiterung für Animationen; wurde nicht verwendet, aber angedacht dies zu tun

Materials - enthält alle Materialien

OSCuMote - Scriptsammlung zur Kommunikation mit einer WiiMote über Ports (Serveranwendung in diesem Projekt -> OSCulator, aber andere möglich)

Prefabs - alle erstellten Prefabs

Resources - alle Ressourcen, wie Audio, Texturen, Meshes etc.

Scenes - alle erstellten Szenen, die in den „Buil Settings“ verwendet werden.

Scripts - alle erstellten Scripts

Standard Assets - Standardverzeichnis für alle importierten Assets

Ressourcen und Werkzeuge

Viele Ressourcen in dieser Semesterarbeit sind aus dem Internet.
Alle Elemente in Unity wurden selbst erstellt (Prefabs etc.).

Für die Terrains wurden entweder Heightmaps (Mond) oder der Terraineditor verwendet. Die Heightmap des Mondes entstand Kartenmaterial der NASA, sowie auch die NASA-spezifischen Meshes. An dieser Stelle sei erwähnt, dass die NASA eine unglaubliche Vielfalt an Ressourcen frei zur Verfügung stellt.
Um die Mondkarte an die eigenen Bedingungen anzupassen wurden die Terrain-Eigenschaften die Unity entsprechend verändert.
Die Landeplattform, sowie der Orientierungspfeil entstand samt Texturierung in Eigenarbeit mit Hilfe von Blender. Auch die Raumfähre, sowie das geladene Mondterrain mussten in Blender angepasst werden, um die Bewegung der Hauptdüse zu realisieren.
Für das anschließende Einladen des Terrains, was nach Bearbeitung in Blender nicht mehr auf einer Heightmap basiert (Terrains in Unity basieren immer auf einer Heightmap), wurden die Editor-Erweiterungen ExportTerrain, Object2Terrain und ObjExporter verwendet.
Benötigte Grafiken für die HUD-Elemente wurden mit Adobe Illustrator erstellt.
Für die Erstellung von Bumpmaps wurde das Programm CrazyBump verwendet, mit dem aus Texturen Bumpmaps erstellt werden können.

Für die Erweiterung von Unity wurden diverse freie Assets verwendet um Spezial-Effekte zu realisieren, wie bspw. die Detonator-Erweiterung. Näheres s. Projektstruktur.

Bekannte Probleme

Der Pfeil der die Richtung anzeigt kann sich bei häufigem Einsatz der Seitentriebwerke stark verschieben und somit starker Veränderung der Rotation der Raumfähre.

Verbesserungsideen

Der Spieler sollte noch mehr motiviert werden, indem gewisse Sachen freigeschaltet werden und ein verstärktes Belohnungssystem eingeführt wird.
Das könnten bspw. Medaillen sein (war für die Endversion geplant, konnte aber nicht rechtzeitig umgesetzt werden).

Die Punkteliste könnte auf einem Server abgelegt werden und für Spieler auf der ganzen Welt sichtbar sein. Das stärkt den Wettbewerbsgedanken. Das würde voraussetzen, dass der Spieler sich vorab mit einer ID anmeldet.

Optische Aufbereitung:
Das Spiel sollte zeitgemäß und hardwareoptimiert dargestellt werden. Das erfordert die Aufbereitung der Meshes (LOD) und die Erwerbung der Pro-Version von Unity, da somit die meisten post-processing-Effekte freigeschaltet werden.
Animierte HUD-Elemente, bspw. für das Kenntlichmachen des Hinzufügens neuer Punkte.

Fazit

Innerhalb dieses Kurses wurde der Umgang mit Unity durch die Erstellung eines Spiels in Form eines Semesterprojekts erlernt.
Die Art mit Unity Spiele zu erstellen macht Spaß und geht schnell im Vergleich zu manch anderer Game Engine.
Dadurch, dass es eine schnelle Arbeitsweise fördert, kann dies aber auch zu chaotischen Projektstrukturen führen.

Ein weiterer Grund für den Autor weiter mit dieser Engine zu arbeiten ist die Portierfähigkeit des Endprodukts auf die bekanntesten Plattformen.

Die Art in Unity Code zu erzeugen war anfangs ungewohnt. Gerade die Kommunikation einzelner Scripte war nicht immer klar.
Mit fortschreitendem Verständnis sind dann nach und nach bessere Alternativlösungen entstanden und das Wissen ein größeres Projekt effizient und performant umzusetzen daraus gewachsen.
Vermutlich reißt die Lernkurve bezüglich der Entwicklung von Spielen mit Unity nie ab, aber mit dem gewonnen Wissen sind evtl. kommende Projekte schneller umgesetzt und „Best-Practice“-Lösungen fließen in diese ein.

Dieser Kurs hat die Instrumentalisierung der Unity Engine, sowie Lösungstrategien für Probleme der Spieleentwicklung, zur Freude des Autors, sehr gut und spannend vermitteln können.
Aufgrund des Thema und der komfortablen und erlebnisreichen Arbeit mit der Unity Engine wurde der Antrieb des Autors hinsichtlich der Fortführung der Arbeit mit der Engine gefördert.

Kurz: „Es hat Spaß gemacht und ich kann es kaum erwarten weiter mit der Engine zu experimentieren.“