VRun - Dokumentation

Medienprojekt

Giuseppe Mario Rizzo - minf100923
Danesh Wohlfart - minf100922

Fachhochschule Wedel, BSc. Medieninformatik

Hamburg, 04.09.2018

Einleitung

Im Rahmen unseres Virtual Reality Projektes haben wir ein Jump & Run Game mit einer HTC Vive Anbindung entwickelt. Die Idee unseres Medienprojektes ist es, an das VR Projekt anzuknüpfen und dieses zu erweitern. Auf der Firmenkontaktmesse **Stuzubi** wurde das Spiel präsentiert. Diese Präsentation bot eine gute Gelegenheit ein Feedback für das Spiel zu bekommen und Verbesserungsvorschläge entgegenzunehmen und zu entwickeln. Hierbei kamen wir zu dem Ergebnis, dass viele Spieler zu Beginn des Spiels überfordert bzw. unsicher mit der Spielmechanik / dem Gameplay sind und es bei den Items und Hindernissen zu Missverständnissen kam.
Die grundlegende Idee des Medienprojektes ist es, diese Missverständnisse zu beseitigen, das Spiel grafisch variabler und spannender zu gestalten und mithilfe einer eigenen kleinen Website das Spiel zu präsentieren und anzuwerben.

Diese Dokumentation fasst nicht mehr die grundlegenden Elemente und die Spielmechanik auf. Dies sollte aus der ersten Version des Virtual Reality Projektes hervorgehen und wird im Sinne des Aufbauprojektes nicht behandelt.
Die Dokumentation zeigt Veränderungen des Projektes auf, erläutert Probleme und erklärt Entscheidungen, die während des Projektes getroffen worden sind.

Inhaltsverzeichnis

1. Verbesserung des Gameplays

Ein Ziel des Medienprojektes ist es, VRUN spannender und variabler für den Nutzer zu gestalten und für ein besseres Spielerlebnis mehr Varianz und Abwechslung in das Spiel zu integrieren. Das Interesse des Nutzers soll damit länger erhalten bleiben.
Die anfängliche, grundsätzliche Idee war es, das Erscheinungsbild der Szene ab einer bestimmten Distanz zu verändern. Erreicht der Spieler einen solche festgelegte Distanz, ändern sich Elemente wie die Umgebung, die Strecke, die Hindernisse oder die Skybox.
Ein Wechsel der Skybox und der Umgebungsobjekte gestaltete sich als kompliziert, verwirrt den Nutzer während des Spiels, und überlädt die Szene mit Logik und Elementen. Daher wurde eine Aufteilung in mehrere Level unternommen, die sich grafisch voneinander unterscheiden und in denen der Nutzer jeweils in eine andere Welt eintaucht. Es wurden 5 Szenen in Unity entwickelt, die sich in Ihrer Erscheinung voneinander unterscheiden:

- HomeScene
  Die HomeScene bildet die Spiellobby, in der sich der Nutzer zu Beginn der Anwendung befindet. Dies ist die einzige Szene in der der Spieler nicht automatisch läuft, sondern auf einem Plateau steht. Von hieraus kann das Spiel gestartet, oder Highscore und Credits eingesehen werden. Es kann ausgewählt werden, ob der Nutzer das Spiel bei Level 1 startet oder ob ein vorhergehendes Tutorial gespielt werden soll.

- Tutorial
  Das Tutorial bietet eine Einweisung in die Spielmechanik und die Interaktionsmöglichkeiten, die dem Nutzer zur Verfügung stehen. Das Tutorial selbst ist ein Level, bei dem der Spieler mehrere Kontrollpunkte erreicht, die chronologisch das Spiel erklären. Weitere Informationen zu dem Tutorial sind in Kapitel '1.8 Erweiterung Tutorial' erläutert. Das Leveldesign ist abstrakt und futuristisch. Alle Elemente wirken geometrisch und modern.

- Level 1
  Das erste Level ist ebenfalls abstrakt gestaltet. Der Nutzer befindet sich, durch das Setzen einer entsprechenden Skybox, in einem Universum und läuft durch Berglandschaften. Objekte, die auf der Spur auftauchen, sind: Kristalle, schwingende Pendel, Zacken (von oben und von unten).

- Level 2
  In dem zweiten Level befindet sich der Nutzer in einem mystischen Wald. Die Skybox ist grün. Es wurde eine dunstige Stimmung erzeugt. Der Nutzer läuft durch einen Wald bestehend aus unterschiedlichen Bäumen. Objekte, die auf der Spur auftauchen, sind: Stachelpflanzen, Pilze, Baumwurzeln und Äste.

- Level 3
  In dem dritten Level befindet sich der Nutzer auf den Osterinseln. Die Skybox stellt einen Himmel dar. Der Nutzer läuft auf einem Deich an den typischen Steinskulpturen vorbei. Objekte, die auf der Spur auftauchen, sind: Steinskulpturen, verschiedene Arten von Felsen

1.1 Spielfluss

Der Spieler befindet sich zu Beginn der Anwendung in der HomeScene. Er hat von dort aus die Möglichkeit das Spiel zu starten, in dem er entweder direkt bei Level 1 startet, oder das Tutorial vorher spielt. Spielt der Spieler das Tutorial durch oder erreicht in Level 1 und Level 2 eine bestimmte festgelegte Distanz, findet ein Szenenwechsel in das nächste Level statt. Level 3 bildet das letzte Level und läuft solange weiter, bis der Spieler verliert.
Verliert der Spieler in einem der Level, erscheint ein *EndScreen* mit dem erreichten Score. Von dort aus wird der Spieler wieder zurück in die HomeScene gesetzt und kann das Spiel von Neuem bei Level 1 beginnen, oder sich die Highscores und die Credits ansehen. Das wiederholte Starten aus dem aktuellen Level ist nicht möglich.

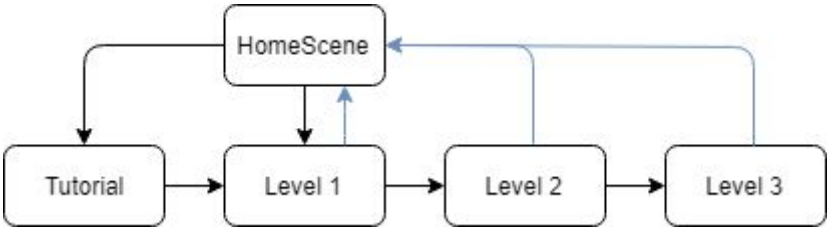


Abbildung 1: Neuer Spielfluss VRun

Schwarze Pfeile: Spielfluss vorwärts
Blaue Pfeile: Wiederkehren zur HomeScene, wenn der Spieler verliert

In der vorherigen Version gab es nur eine Szene und der Spieler hatte die Möglichkeit nach der Kollision mit einem Objekt wieder von vorne zu beginnen. Die Highscore-Tabelle war in den *EndScreen* integriert.

Abbildung 1.2: Alter Spielfluss von VRun
Blauer Pfeil: Erneutes Spielen nach Kollision mit einem Hindernis

1.2 Szenenwechsel

Der Szenenwechsel wurde durch ein Portal realisiert, welches ab einer bestimmten festgelegten Distanz auf der Spur platziert ist. Durch ein schwarzes Partikelsystem wird dem Nutzer suggeriert, dass er in eine Finsternis läuft. Sobald der Spieler durch das Portal läuft, wird vor dem Spieler eine schwarze Fläche eingeblendet, die das Sichtfeld verdeckt. In dieser Zeit wird die neue Szene aktiviert, die zuvor asynchron geladen wurde. Ist die neue Szene aktiviert, wird die schwarze Fläche ausgeblendet und der Spieler befindet sich, weiterhin laufend, in dem neuen Level. Dem Spieler wird suggeriert, dass er in die Finsternis läuft und aus dieser auch wieder herausläuft und plötzlich sich in einer anderen Welt befindet.

1.3 Erweiterung der HMD Anbindungen

In der vorherigen Version von VRun wurde zur Einbindung der HTC Vive das Virtual Reality Toolkit [1] verwendet. Dieses Toolkit wurde durch das, öffentlich zur Verfügung stehende, VRSF [2] ersetzt. Diese Einbindung hat den Vorteil, dass sowohl eine HTC Vive Anbindung als auch eine Oculus Rift Anbindung möglich ist. Das Spiel kann jetzt mit der HTC Vive oder der Oculus Rift gespielt werden. Ist keine HMD angeschlossen aktiviert sich ein Simulator und das Spiel ist auch mit der Tastatur spielbar. Dies ist für das Spielgeschehen nicht vorgesehen. Mit dem Simulator kann man durch die komplette Szene fliegen und den Spielbereich verlassen. Im Sinne des VR Konzeptes und aufgrund mangelnder Zeit wurde eine Einschränkung der Bewegung nicht weiter behandelt.

1.4 Erweiterung der Nutzeraktionen

Die Spielmechanik und die Interaktionen in Bezug auf das endlose Laufen sind bis auf einen Aspekt gleich geblieben. Es ist nun zusätzlich möglich mit den Controllern die Coins einzusammeln. Die Controller wurden um einen Collider erweitert, der nur bei den Punkten eine Kollision auslöst. Befindet sich der Spieler auf der Spurmitte, kann er durch das Schwenken der Controller nach rechts und links auch Coins einsammeln, die neben ihm platziert sind. Auch diese Idee ließ sich auf das Testen des Spiels bei der Stuzubi-Messe zurückführen. Die meisten Spieler nahmen fanden es intuitiv die Coins auch mit den Controllers einzusammeln und wunderten sich, dass dies nicht möglich war.

1.5 Verbesserung der Spielerkollisionen

Auf dem Spieler befindet sich ein *Capsule Collider*, der die Kollisionen mit den Hindernissen veranlasst. In der vorherigen Version war der Collider an das GameObject gebunden, welches den Kopf des Spielers darstellt. Auf der Stuzubi-Messe fiel dabei eine Problematik auf. Der *Capsule Collider* rotiert bei Neigung des Kopfes mit. Hat der Spieler seinen Kopf stark nach rechts oder nach links geneigt, neigt sich auch der Collider und befindet sich dabei quer über der Spur. Dies hat zur Folge, dass sich der Collider über mehrere Spuren legte und eine Kollision mit einem Hindernis stattgefunden hat, obwohl der Spieler mit seinem Körper ausgewichen ist.

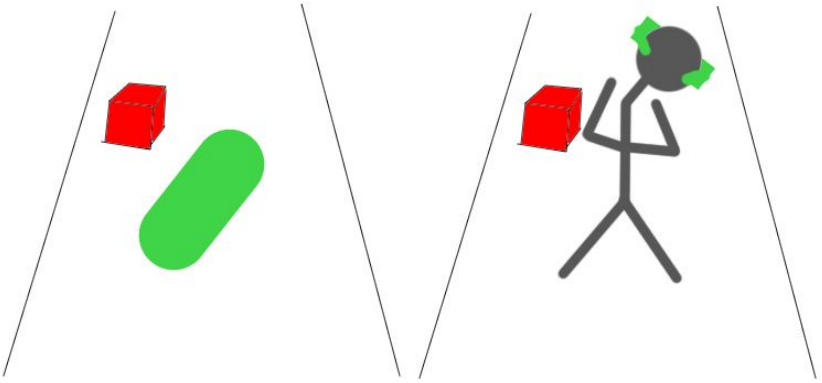


Abbildung 2: Visualisierung des Capsule Colliders mit Rotation

Aus diesem Grund wurde der Capsule Collider so modifiziert, dass dieser nicht die Rotationen der Brille übernimmt und immer vertikal auf der Spur bleibt. Es findet lediglich eine Positionsveränderung bezüglich der x-, y-, z-Achse statt.

1.6 Objektspawning

In der vorherigen Version von VRun wurden zu Beginn des Levels mehrere gleiche Objekte hintereinander platziert, sodass ein Objektfeld entsteht. Diese objektspezifischen Felder wurden nacheinander gespawnt und sollten dem Nutzer eine Einführung und Gewöhnung an die Spielobjekte bieten.
Das Objektspawning findet nun pro Level komplett zufällig statt. Es werden keine Objektfelder mehr erstellt. Das hinzugefügte Tutorial zeigt auf, welche Objekte von dem Spieler eingesammelt werden können. Auf eine Einführung innerhalb der Level kann somit verzichtet werden und steigert aufgrund der höheren Objektvarianz zu Beginn der Level das Interesse des Spielers.

1.7 Einbinden von Physics

Für die Erweiterung des Spiels wurde entschieden, dass eine im letzten Projekt geplante kurvige Strecke nicht implementiert wird. Die Bezier Berechnung wurde verworfen und stattdessen eine Unity typische Vorwärtsbewegung abhängig von Bewegungsvektor, Geschwindigkeit und `Time.fixedDeltaTime` implementiert.
Es kam die Idee auf als weiteres Hindernis Lücken in die Spur zu integrieren über die der Nutzer springen muss. Gelingt ihm dies nicht, soll er von dem Track fallen. Ein Fallen von der Spur wurde mithilfe der Unity Physics integriert. Dabei traten mehrere Probleme auf. Das Fallen des Spielers in VR führte zu einem Unwohlbefinden und zu starker motion sickness. Das Prinzip des Einblenden des Endscreens konnte auch nicht intuitiv gelöst werden, da der Spieler im Fall nicht stoppt. Ein plötzliches Stoppen würde wiederum unnatürlich wirken und zu Verwirrung führen. Hindernisse in Form von Lücken in der Spur wurden nicht hinzugefügt.

1.8 Erweiterung Tutorial

Das Tutorial soll dem Spieler eine Einführung in die möglichen Interaktionen und die Spielmechanik geben. Bei der Präsentation des Spiels auf der Stuzubi-Messe stellte sich heraus, dass das Spiel und die Spielmechanik nur mit einer persönlichen Einführung verständlich wurden. Daher wurde das Spiel in dieser Version um ein Tutorial erweitert.

In der HomeScene hat der Nutzer die Möglichkeit das Spiel direkt mit Level 1 zu beginnen, oder alternativ das Tutorial zu starten. Diese Möglichkeiten sind durch 2 Buttons realisiert worden.

Das Tutorial besteht, wie das normale Spiel auch, aus einer Strecke auf der Objekte platziert sind. Es hebt sich dadurch ab, dass der Spieler nicht endlos läuft, sondern von einem Kontrollpunkt zu dem nächsten. Jeder Kontrollpunkt erklärt eine bestimmte Eigenschaft des Spiels. Auflistung der Kontrollpunkte:

1. *Willkommenstext*
2. *Links und Rechts Bewegungen*
3. *Springen und Ducken*
4. *Das Punktesystem*
5. *Die Sammeln von Coins*
6. *Das Benutzen von Items*

Misslingt dem Spieler eine Aktion, die in dem aktuellen Kontrollpunkt beschrieben ist, wird der Spieler wieder an diesem Kontrollpunkt gespawnt und kann die Aktion wiederholen. Nach Beenden des Tutorials beginnt der Spieler automatisch das Spiel und wird in Level 1 gespawnt.

1.9 Konzeption und Leveldesign

Das Leveldesign, die Konzeptionen und die Ideen der Level und der darin befindlichen Objekte wurden von uns entwickelt. Alle in dem Spiel befindlichen 3D Objekte wurden von einer befreundeten, externen Person entwickelt, die unsere Ideen umgesetzt hat und uns die 3D Objekte frei zur Verfügung stellte. Diese wurden von uns in das Spiel eingebunden, platziert und um die jeweiligen Interaktionen ergänzt.
Die Hintergrundbilder der UI Elemente wurden mittels Adobe Illustrator von uns selbst erstellt.
Die verwendeten Skyboxen, sowie die Titel der Hintergrundmusik sind lizenzfrei. Verweise auf die dahinter stehenden Künstler werden in den Credits in der HomeScene gelistet.
Die Hindernisse, die Spur, die Umgebungsobjekte und die Skybox sind levelspezifisch. Um den Nutzer, bezüglich der Items, nicht zu verwirren und eine klare Unterscheidung zwischen Hindernisse und Items zu treffen, bleiben die Item-Objekte und auch die Coins levelübergreifend gleich. Die Coin-Modelle wurden zu Früchten verändert.
Jede Szene wurde zusätzlich durch eine ausgewählte, zu der Welt passenden, Hintergrundmusik erweitert. Bei der Implementation fiel auf, dass die Hintergrundmusik, neben den visuellen Aspekten der Szenerie, stark die Immersion beeinflusst und einen wesentlichen Teil zu dem Spielspaß-Faktor beiträgt.

1.10 Auflistung hinzugefügter Elemente

- General
    - Die Coins wurden durch Früchte ersetzt
    - Neuer EndScreen ohne Highscore-Funktion
    - Ein Portal wurde den Level 1 -3 hinzugefügt
    - Jede Szene unterliegt einer anderen Hintergrundmusik

- HomeScene
    - VR Canvas-Main Menu
    - VR Canvas-Score
    - VR Canvas-Credits
    - Plattform

- Tutorial
    - 7 VR Canvas - ControlPoint
    - 2 verschiedene abstrakte Hindernisse (Ausweichen links/rechts, oben/unten)
    - szenenspezifisches Portal
    - szenenspezifisches Trackelement : abstrakte Bodenerhöhung

- Level 1
    - szenenspezifisches Trackelement: dunkle Kacheln

- Level 2
    - Hindernisse: Stachelpflanze, Pilz, Duckast, Sprungranke
    - szenenspezifisches Trackelement: Holzdielen
    - Hintergrund: grüne Skybox
    - Umgebungsobjekte:
      Wald (bestehend aus einer Anhäufung von 4 verschiedenen Bäumen)
      Laternen über der Spur

- Level 3
    - Hindernisse: Skulpturen, Felsen, Sprungfelsen
    - szenenspezifisches Trackelement: Bodenerhebung (Deich)
    - Hintergrund: Himmel - Skybox
    - Umgebungsobjekte: Skulpturen

2. Verbesserung der Codebasis

Das Spiel wurde auf die neuere Unity Version 2018.2.6 aktualisiert. Im Zuge der Spielerweiterung und durch das Hinzufügen von mehreren Szenen, mussten die Spieldaten modular so verwaltet werden, dass alle relevanten Spielinformationen und -daten trotz eines Szenenwechsels zur Verfügung stehen. Dazu wurde die gesamte aufgebaute Architektur überarbeitet und an den neuen Anwendungsfall angepasst. Wesentlich wurden zwei Veränderungen vorgenommen. Statt, der durch Unity zur Verfügung stehenden Referenzen, die im Editor gesetzt werden können, wurde ein durch C# definiertes Event-System implementiert. Zusätzlich wurden die bestehenden *GameObjects* des vorherigen Spiels neu strukturiert und modularisiert.

2.1 Event System

Unity bietet die Möglichkeit mittels Referenzen, die im Editor gesetzt werden können, Skripte miteinander zu verbinden, um aus einem Skript Zugriff auf die öffentlichen Funktionen/Variablen des referenzierten Skripts zu bekommen.
Mit zunehmender Projektgröße wurden diese Referenzierungen unübersichtlich und schwer nachvollziehbar für das andere Projektmitglied. Bestimmte Spielaktionen erfordern in unterschiedlichen Skripten und unterschiedlichen semantischen Bereichen Funktionsaufrufe.

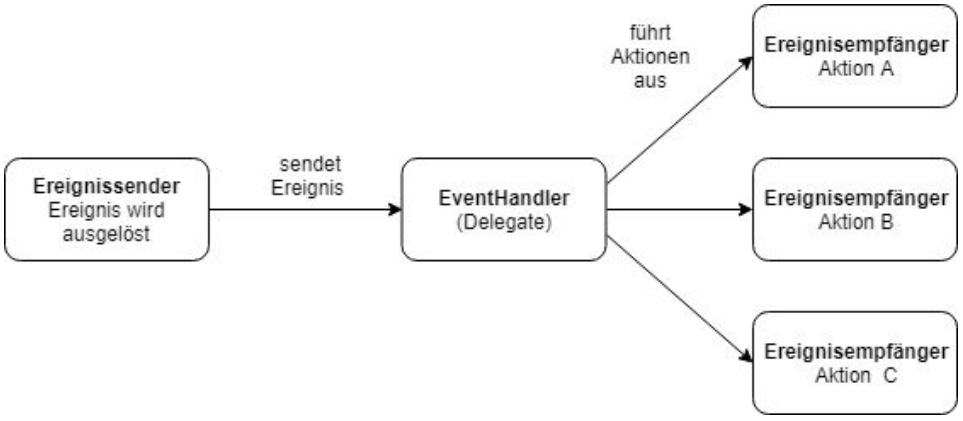
Folgendes beispielhaftes Szenario:
Der Spieler kollidiert mit einem Hindernis. Aus dieser Spielaktion gehen mehrere Aktionen hervor, die ausgeführt werden müssen:

- Der Punktestand muss abgespeichert werden
- Der EndScreen muss initialisiert werden
- Der Kollisionssound muss abgespielt werden
- Der Spieler muss stoppen
- Ein möglicher aktivierter Timer muss beendet werden

Das Setzen jeder dieser Skriptreferenzen führt zu Unübersichtlichkeiten. Außerdem ist wiederholt aufgetreten, dass bei der Verwendung der Versionskontrolle Referenzen verloren gehen und neu hinzugefügt werden müssen.

Um dem Setzen von Referenzen aus dem Weg zu gehen, kam die Idee auf ein Event-System zu implementieren. An beliebiger Stelle kann ein definiertes Event aufgerufen werden und alle dem Event zugewiesenen Funktionen werden daraufhin ausgeführt.
Ein solches Event-System lässt sich zum einen über C# mit *Ereignissen und Delegaten* realisieren und zum Anderen durch Unity spezifische *Scriptable Objects*, die innerhalb des Editors definiert und benutzt werden können.
Die Möglichkeit innerhalb des Unity Editors zu Arbeiten und Referenzen zu setzen oder *Scriptable Objects* zu erstellen, bietet eine Reihe von Vorteilen. Unity ist eine Plattform mit der Spiele entwickelt werden können. Neben Programmierern wirken bspw. Grafiker oder Konzeptdesigner an den Projekten mit. Der Unity Editor bietet die Möglichkeit, mit zur Verfügung stehenden Skripten, ohne Kenntnisse in der Programmierung Spiele bausteinartig ”zusammen zu klicken”. In einer Teamarbeit können Programmierer Skripte entwickeln und zur Verfügung stellen, die dann von anderen Bereichen lediglich eingebunden werden können.
Das Beispiel des Event-Systems zeigt auf, dass an einigen Stellen von den Entwicklern selbst eine Entscheidung getroffen werden kann, inwiefern man die zur Verfügung stehenden Implementationen von Unity verwendet.
In diesem Projekt wurde das Event-System mittels den C# spezifischen *Ereignissen und Delegaten* realisiert und auf die Unity spezifischen *Scriptable Objects* verzichtet.

Ablauf eines Ereignisses



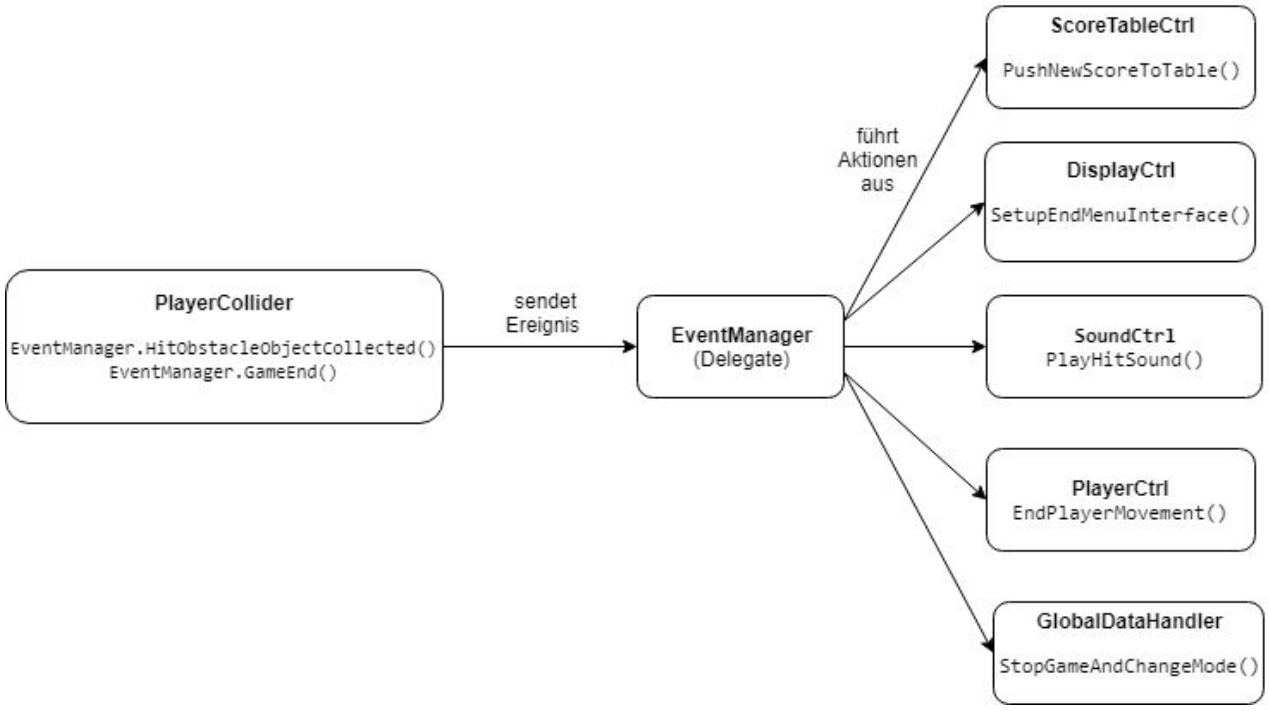
Innerhalb des *EventManager* Skripts werden folgendermaßen alle Ereignisse deklariert.

- Ein Delegat wird definiert, welches im Kontext der Ereignisse, wie ein Funktionszeiger agiert. Es muss in seiner Signatur denselben Rückgabewert zurückgeben und die gleichen Parameter empfangen, wie die Methode, die durch ein Ereignis aufgerufen werden soll.

- Eine `event` Variable wird im Eventhandler deklariert, die alle Methodenreferenzen speichert, die beim Senden des Ereignisses aufgerufen werden sollen und das Ereignis damit abonnieren. Die Abonnenten werden Ereignisempfänger genannt. Innerhalb der Ereignisempfänger müssen die Methoden bzw. Aktionen der `event` Variable hinzugefügt werden. Dies geschieht in den jeweiligen Skripten durch die `OnEnable()` Funktion

- Zum Senden eines Ereignisses wird eine `public static void` Methode deklariert, die aufgerufen werden soll, wenn ein Ereignis gesendet werden soll.

Bezogen auf das genannte Szenario ergibt sich folgende Struktur:



2.2 Modularisierung

Durch das Hinzufügen mehrerer Szenen ist eine Modularisierung und eine gute Strukturierung der Daten einer der Kernaspekte bei der Erweiterung des Spiels gewesen. Grundlegend lässt sich die Spielstruktur und der Spielfluss in 2 Elemente aufteilen:

2.2.1 Szenenspezifische Elemente

Diese Elemente sind Unity definierte *GameObjects* bzw. *Prefabs* mit den dazugehörigen Skripten, die für die Logik und die richtige Darstellung einer speziellen Szene zuständig sind. Jede Szene besitzt die folgenden Prefabs, die für die Spiellogik essentiell sind:

- TrackController
  Der *TrackController* ist für das Generieren der Streckenabschnitte zuständig auf denen der Spieler läuft. Hierbei werden Streckenelemente vor und hinter dem Spieler jeweils dynamisch gelöscht bzw. generiert.

- DisplayController
  Der *DisplayController* ist für die Anzeige der Spielinformationen zuständig. Angezeigt werden der Punktestand, die Restzeit und ein EndScreen, der bei einer Kollision mit einem Hindernis erscheint.

- ObjectList
  Die *ObjectList* ist für das Spawnen der Hindernisse, Items und Coins auf der Strecke zuständig. Hierbei werden Hindernisse, Items und Coins vor und hinter dem Spieler jeweils dynamisch gelöscht bzw. generiert.

Neben den hier genannten *GameObjects* können die einzelnen Szenen durch weitere *GameObjects* in der *Hierarchy* individuell erweitert werden. Level 1 besitzt einen zusätzlichen *MapController,* der die Berglandschaften generiert. Dies wurde in Level 2 & 3 durch ein weiteres *TrackCtrl* Skript im *TrackController* realisiert. Zusätzlich befindet sich zu Beginn in Level 1 ein statischer Tunnel, der aus der alten Implementation übernommen worden ist.

2.2.2 Szenenübergreifende Elemente

Szenenübergreifende Elemente sind GameObjects, die bei einem Szenenwechsel von der aktuellen Szene in die neue Szene übernommen werden. Diese Elemente beziehen sich zum einen auf den Spielstatus und enthalten Informationen über das aktuell laufende Spiel (wie bspw. der Punktestand) und zum anderen bieten sie die Schnittstelle der HMD zum Spiel. Sie können als Instanz verstanden werden, die über den ganzen Spielablauf bestehen bleiben und jeweils in die aktuelle Szene geladen werden.

- GameManager
  Der *GameManager* bildet das zentrale GameObject, welches alle Daten und damit den Zustand des Spiels speichert und verarbeitet. Der *GameManager* wird in Kapitel *2.2.2.1 GameManager* erläutert.

- *CameraRig
  Das *CameraRig* bietet die Schnittstelle zwischen der HMD und dem Spiel und bindet die Controller und die Brille als GameObjects in die Szene ein. Dieses Prefab kann als Spielerinstanz verstanden werden. Skripte wie der *PlayerCollider* und der *PlayerCtrl* sind hier als Komponenten hinzugefügt.
  (, da die Bezeichnung bei unterschiedlichen HMD Typen oder Simulator variiert)*

- SetupVR_*
  Das SetupVR_* enthält als Komponenten alle notwendigen Skripte, welche die VR-Interaktionen zur Verfügung stellen. Es ist der Teil der VRSF Implementierung.
  (, da die Bezeichnung bei unterschiedlichen HMD Typen oder Simulator variiert)*

2.2.2.1 GameManager

Durch die Erweiterung des Spiels um mehrere Szenen stellte sich die Frage wie die Spielinformationen wie bspw. die zurückgelegte Distanz oder Spielparameter wie z.B. die Spielergeschwindigkeit so festgelegt werden können, dass sie in jeder Szene zur Verfügung stehen und global zugreifbar sind.

Der GameManager ist ein GameObject, welches mehrere Skripte als Komponenten einbindet, die die wesentlichen Informationen über das aktuelle Spiel speichern und verwalten und die außerdem den Spielfluss bezüglich der Nutzerinteraktionen kontrollieren. Der GameManager bildet damit ein zentrales Element des gesamtes Spiels. Er ersetzt den *GameController* aus der vorherigen Version. Die Instanz des GameObjects wird zusammen mit den anderen szenenübergreifenden Elementen in die jeweilige Szene übergeben, sodass alle Spielinformationen in jeder Szene zur Verfügung stehen.

Der GameManager besteht hält folgende *Components:*

- EventManager
  Das *EventManager* Skript deklariert alle Ereignisse, die durch das Event-System aufgerufen werden können.

- GlobalDataHandler
  Das *GlobalDataHandler* Skript legt alle Spielinformationen und Spielparameter fest, die das Spielgeschehen definieren und beeinflussen. Dazu gehören:

  > Entfernung der Meilensteine
  > Startgeschwindigkeit
  > Geschwindigkeitserhöhung
  > Debug-Modus aktiviert (Flag)
  > Streckenbreite
  > Spawnpositionen der Objekte
  > RunMode (Flag, ob der Spieler fortlaufend rennt)

  > aktuelle Spielergeschwindigkeit
  > Grundgeschwindigkeit
  > Kollisionsstatus
  > Punkteverdoppler aktiviert (Flag)

  > Itemtypen
  > Bewegungsmodus

  Das Skript stellt hierfür Getter- und Setter-Funktionen zur Verfügung, die den Zugriff und das Abrufen dieser Werte ermöglichen.

- InputManager
  Der *InputManager* verarbeitet alle Tastatureingaben. Die Tastatureingaben sind für den Spieler nicht relevant. Die Eingaben auf der Tastatur wurden zu Testzwecken eingebunden und bildeten vor der VR-Implementation die grundlegende Spielsteuerung. Eine Erklärung der Tastenbelegung befindet sich in Kapitel *3.3 Tastatureingaben*.

- SoundCtrl
  Das *SoundCtrl* Skript bindet alle Audio-Dateien und Soundeffekte in das Spiel ein und stellt Funktionen zur Verfügung, die ausgewählte Sounds abspielen. Diese Funktionen werden durch das Event-System aufgerufen.

- LevelManager
  Das *LevelManager* Skript stellt Funktionen zur Verfügung, die bei den Szenenübergängen aufgerufen werden und die für das Laden der neuen Szene zuständig sind. Diese Funktionen werden durch das Event-System aufgerufen.

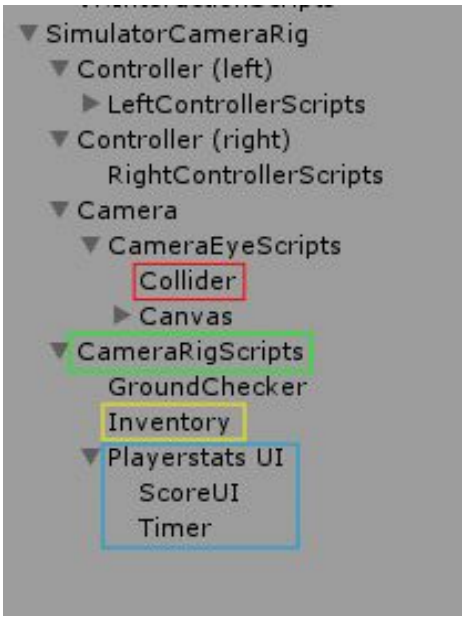
- ScoreTableCtrl
  Das *ScoreTableCtrl* Skript verwaltet die Highscore-Tabelle und ist für das Aktualisieren, das Laden und das Speichern der Tabelleneinträge zuständig. Die hierbei entwickelten Funktionen werden durch das Event-System aufgerufen.

2.2.2.2 Die Spielerinstanz

Im Vergleich zu der vorherigen Version wurden einige Komponenten klarer voneinander getrennt. Besonders die Spielerinstanz wurde modularer aufgeteilt. Die Bewegung des Spielers werden durch das *PlayerCtrl* Skript gesteuert und sind nun komplett vom *TrackCtrl* Skript getrennt. Es ist ein eigenes *InventoryHandler* Skript für den Inventar und ein *ScoreHandler* Skript für den Punktestand entstanden, die bei Bedarf einfach de-/ aktiviert werden können. Kollisionen werden durch das *PlayerCollider* Skript verarbeitet.

Alle aufgeführten Skripte sind Komponenten, die in das von VRSF zur Verfügung gestellte *CameraRig eingebunden worden sind. Zusätzlich wurden die UI Elemente der Punktzahl und des Timers für das eingesammelte Item hinzugefügt, da sich diese Elemente mit dem Spieler durch die Szene bewegen und abhängig von dessen Position sind.

Komponentenzuweisung der Skripte zu den *GameObjects:*



Collider (rot):	*PlayerCollider*
CameraRigScripts (grün):	*PlayerCtrl*
Inventory (gelb):	*InventoryHandler, ScoreHandler*
Playerstats UI (blau):	*UI Elemente Score, Timer*

*Abbildung 3: Unity Hierarchy - *CameraRig der Don't Destroy On Load Instanz*

3. Bedienungsanleitung

3.1 Einrichtung

VRUN kann mit der HTC Vive und der Oculus Rift gespielt werden und benötigt ein eingerichtetes Computer-Setup. Entsprechende Hardwareanforderungen und Anweisungen zur Inbetriebnahme sollten aus den offiziellen Dokumentation entnommen werden. Die VR Anbindung wurde in diesem Projekt mit SteamVR durchgeführt. Es sollte darauf geachtet werde, dass beide Controller aktiviert sind und erkannt werden. Andernfalls kann es zu Problemen bei der Steuerung kommen.

Das Spiel erkennt automatisch welche HMD angeschlossen ist. Es müssen keine extra Einstellungen vorgenommen werden. Wird das Spiel ohne eine angeschlossene und eingerichtete HMD gestartet, ist das Spiel über den VRSF Simulator spielbar. Dieser wurde für Testzwecke verwendet, bietet aber aufgrund der in Kapitel *1.3 Erweiterung der HMD Anbindung* erwähnten Bewegungsmöglichkeiten keine originale Spielmechanik.

3.2 Spielbeginn

Zum Starten des Spiels sollte die vorliegende VRUN.exe Datei ausgeführt werden. Der Nutzer befindet sich anfangs in der Spiellobby und kann mithilfe des rechten Controllers und dem darauf befindlichen Laser auf die Schaltflächen *Play* oder *Tutorial* zielen und diese durch einen Druck auf die (in beiden HMDs definierten) Trigger Buttons des rechten Controllers aktivieren. Ist der Laser auf den entsprechenden Button gerichtet, ändert sich die Laserfarbe.

Eine Anleitung des Spielverlaufes, der -mechanik und der -aktionen ist durch das spielbare Tutorial gegeben und wird außerdem in der vorherigen Version des VR-Projektes erläutert.

3.3 Tastatureingaben

Der Simulator überträgt die Steuerung der Controller auf die Tastatur. Folgende Belegung wird dabei verwendet:

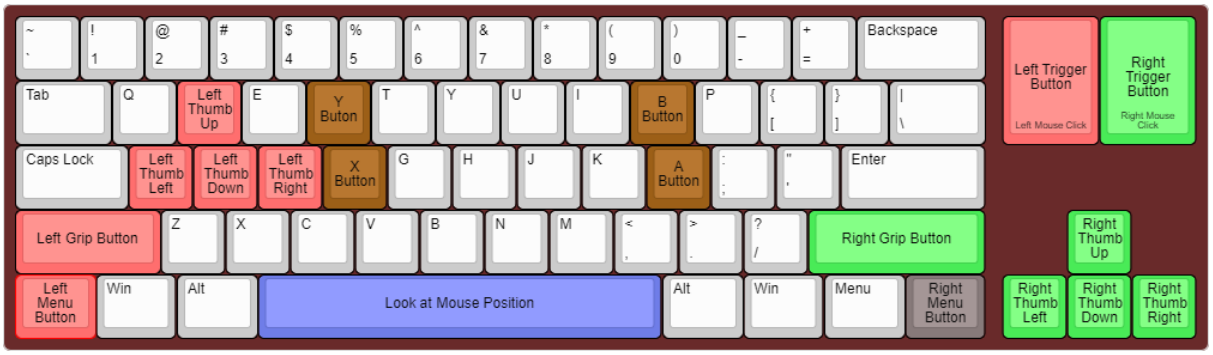


Abbildung 4: Tastaturbelegung des VRSF Simulators, Quelle: [3]

Die Tastenbelegungen des InputManagers, welche zu Testzwecken verwendet worden sind, sind folgendermaßen definiert:

Taste	Beschreibung
g	Startet das Spiel
r	Beendet das Spiel und löst einen Reset aus
c	De-/Aktivierung des Kollisionsmodus mit Hindernissen
v	Aktiviert das eingesammelte Item
w	Veranlasst das Weiterlaufen bei einem Kontrollpunkt im Tutorial
k	De-/Aktiviert die Sounds innerhalb des Spiels
Esc	Beendet das laufende Spiel und spawnt den Spieler wieder in der HomeScene

Tabelle 1: Tastenbelegung für Testzwecke

4. Veröffentlichung

4.1 Erstellen einer Webseite

Zur Vorstellung des Spiels und im Sinne der Veröffentlichung wurde eine Webseite entworfen, die das Spiel kurz präsentiert und einen Download-Link zur Verfügung stellt.
Die Website ist unter folgendem Link aufrufbar:

https://vrun-game.github.io/VRUN/

Die Webseite liegt in Form einer Onepager - Website vor und bietet einen kurzen Erklärungstext, der beschreibt was VRUN für ein Spiel ist. Zu jedem Level gibt es einen Screenshot und einen kurzen Beschreibungstext.

Zuletzt befindet sich ein Download-Link des Spiels auf der Webseite, mit dem es möglich ist das Projekt direkt herunterzuladen.

4.2 Veröffentlichung des Projektes

Die Umstrukturierung und das Hinzufügen der neuen Spielelemente nahm mehr Zeit und Aufwand in Anspruch als in der Planung festgelegt worden ist. Es wurde zu Beginn recherchiert auf welchen Plattformen das Spiel möglicherweise publiziert werden könnte. Folgende Ergebnisse sind dabei entstanden:

- Steam
- HumbleBundle
- UPlayPC
- itch.io
- gog.com
- greenmangaming.com
- gamersgate.com

Aufgrund des Zeitmangels wurden diese Möglichkeiten nicht weiter in Erwägung gezogen und das Spiel auf GitHub veröffentlicht. Das Spiel ist unter dem folgenden Link abrufbar:

https://github.com/vrun-game/VRUN

5. Links

[1] https://vrtoolkit.readme.io/
[2] https://github.com/Jamy4000/VR-Scriptable-Framework
[3] https://github.com/Jamy4000/VR-Scriptable-Framework/blob/master/Wiki_Resources/Keyboard_Simulator_Mapping.png