Konzeptdokumentation von Erwin Oudomvylay

Ideenfindung
Als Themenbereich sind wir in Richtung Bildung gegangen und haben uns für das Fach Physik entschieden. Mit der VR Brille kann man jegliche Simulationen testen, die in der Realität fast unmöglich sind. Der Fokus dabei liegt am Gravitationsunterschied auf verschiedenen Planeten in unserem Sonnensystems.

Als Simulation wurde Basketball als Sportart ausgewählt. Nebenbei gibt es ebenfalls Dosen- und Dartwerfen als Tests auf dem Raumschiff.

Zielgruppe
Unsere Zielgruppe entsprechen Schüler/innen der 10. Klasse. Ein Teilbereich der Physik beschäftigt sich mit den Grundprinzipien der Gravitation.

Story
Eine Zivilisation außerhalb der Erde entdeckt Basketball als Sportart, die einst vor vielen Lichtjahren als Signal auf deren Heimatplanet empfangen wurde. Jetzt suchen sie nach einem Planeten, was die perfekten, physikalischen Konditionen besitzt, wo man die intergalaktische Basketball Meisterschaft spielen könnte. Jedoch ist der Player nicht alleine: Man hat eine AI Voice als Companion, die den Player auf der Suche nach dem richtigen Planeten behilflich ist.

Controls & Locomotion
Als Locomotion wurde für den Teleport entschieden und als Turn der Snap turn.
- Linker Joystick: Beim Halten nach vorne wird ein Teleport Ray angezeigt und beim Loslassen wird man zum Endpunkt teleportiert
- Rechter Joystick: Nach links / rechts wird man in die jeweilige Richtung um 45° gesnappt und nach unten geflickt wird der Player um 180° gedreht
- A / X Taste: Ein Ray wird aus der Hand ausgestrahlt, worin man nur mit den Buttons auf der UI interagieren kann
- Trigger Taste: Beim Drücken des Trigger Buttons mit dem Ray wird die Funktion des jeweiligen Buttons ausgeführt; Allgemein spielt dies eine Pinch Animation ab und man kann damit ebenfalls mit der UI interagieren.
- Grip Taste: Damit kann man Objekte greifen
- Y Taste: Man kann das Menü umschalten
- B Taste: Die Position des Menüs wird vor die Player Cam repositioniert

Hand Model
Das offizielle Handmodell von der Meta Quest 2 wurde hierfür genommen und die schon vorhandenen Animationen verwendet. Es gibt eine Greif-Animation und eine Pinch-Animation.

Physics Material
Basketball:
Jedes Basketball besitzt die gleiche Bounciness Wert unabhängig von der Atmosphäre. Die Static und Dynamic Friction spielt in der Realität eine große Rolle, da dies vom hohen Druck der Planetenatmosphäre beeinflusst wird. Jedoch spielt der Wert in der Simulation kaum eine Rolle, da dies sehr niedrig wäre und man es kaum merken würde, da Unity in VR die physikalische Atmosphäre nicht wahrnehmen kann.

Field:
Das Basketballfeld wurde für den Basketball angepasst, damit es gescheit vom Boden bounct.

Tennisball:
Die Tennisballwerte wurde für einen normal Tennisball angepasst.

Input Manager
Ein eigener Input Manager wurde hierfür erstellt, um das Menü umzuschalten und zu repositionieren.

Teleport Anchor
Der Teleport Anchor soll als Hilfestellung darbieten damit man, anhängig von der Anchor Position, in die richtige Richtung schaut, wenn man zum Ankerpunkt teleportiert.

UI Positionierung
Die Positionierung der UI war sehr umständlich, wie wir es jetzt letztendlich implementieren wollten.
Am Anfang stände es an fixe Positionen, jedoch tut man diese an machen Perspektiven sehr schwer sehen.
Dann wurde für eine Hand Menü überlegt und implementiert, wenn man die Bewegung machen würde als würde man auf die Armbahnuhr schauen, dann erscheint das Menü. Hier war das Problem, dass man dann nicht mehr die Objekte greifen konnte, wo das Menü war. Auf der linken Hand konnte man sich gar nicht mehr teleportieren.
Danach gab es eine Update Methode, was die Menüposition an die Cam Position kotinuierlich mit einem Offset ankettet. Das Problem hierbei war, dass wenn man den unteren Rand der Menus sehen wollte, war dies sehr schwer, da sich das Menü bei der Kopf mitbewegt.
Somit musste es bei Knopfdruck die UI vor dem Offset der Cam gesetzt werden. Zum Schluss wurde dann für das Repositionieren und das Umschalten des Menüs vor die Cam entschieden, was die UI viel angenehmer macht.

Light Probe
In jeder Szene ist eine Light Probe vorhanden, um dynimasche Objekte baken zu lassen.

Hierarchie
Dynamic: Alles, was sich in der Szene bewegt.
Environment: Die Skydome.
Interface: Die komplette VR UI Interaktion.
Lighting: Reflection Probe und Lighting.
Manager: Alle Manager Skripte.
Spawn: Alle Positionen der Spawnpunkte.
Static: Alle Objekte, die sich nicht bewegen. Sie werden auf static gestellt für das Lighting.
XR: Die komplette XR Steuerung, bestehend aus einem Poke, Ray, Direct und Teleport Interactor.
Sound: Musik und Ambience für die jeweiligen Szenen.

Scripts

AudioManager
Es wurde als Simpleton instanziiert da jedes Script darauf zugreifen kann, da hier die Audio behandelt wird.
Der voiceLines Array muss im Inspektor chronologisch in der richtigen Reihenfolge zugeordnet werden. Dies ist abhängig von der PlayVoice() Methode, da der voiceIndex mit jedem Aufruf inkrementiert wird.

Basketball
Zunächst wird im Awake der Rigidbody Threshold, abhängig von der Szene, eingestellt. Dies hat den Zweck, dass ein Basketball, wenn es fällt, am Ende immer schneller auf dem Boden ankommt bis es sich nicht mehr bewegt. In Unity war dies recht schwer zu lösen; wenn man den Default Threshold genommen hat, war der Basketball auf der Erde und auf dem Mond passend, jedoch auf dem Jupiter, weil die Gravitation sehr stark ist, bounct der Basketball schnell und der Ball würde nie zum Schlafen kommen sondern kontinuierlich bouncen. Auf dem Jupiter sind die Threshold recht hoch eingestellt und auf dem Mond dann dementsprechend sehr niedrig. Die Werte werden je nach buildIndex angepasst.
(Default Bounce Threshold Value: 2.0f; Sleep Threshold: 0.005f;)

TaskOne: Dribbeln
Für die Dribbeln Logic muss der Player den Ball 3x hintereinander auf dem Boden dribbeln und in der Zeit, wo der Ball den Boden berührt und wieder hochkommt, nochmal greifen und wieder zum Boden werfen, damit der Score hochgeht.
Wenn man zweimal den Boden berühren lässt, dann resettet der Score zu 0 und man muss von vorne anfangen.
Auf dem Jupiter ist die Schwierigkeit beim Dribbeln sehr hoch und es wurde für kein Failsafe nachgedacht (verglichen zum Korb werfen), weshalb der Player in die Hocke gehen oder sitzen muss, um den Ball vom Boden aus greifen zu können.

TaskTwo: Korbwerfen
Der Basketball muss in dem Korb geworfen werden, damit man zurück zum Schiff kann.
Mit jedem fehlgeschlagenem Versuch (wenn der Basketball den Boden berührt) sinkt der Korb um etwas nach unten, da dies sonst unmöglich auf dem Jupiter wäre (Das ist der Failsafe).
Um jedoch den Korb immer zu sinken, muss man einen neuen Ball spawnen und werfen, da man sonst die Mechanic ausnutzen könnte, in dem man immer auf dem Boden wirft. Wenn man auf Nein klickt, ob man zurück zum Schiff will, muss man wieder in den Korb reinwerfen, um zurück zu gelangen.

ButtonHandler
Alle Buttons besitzen dieses Skript. Es gibt drei Haupttags: TaskOne und TaskTwo und VoiceClip.
Bei TaskOne bzw. TaskTwo wird der dazugehörige GameState im GameManager geändert.
Alle Buttons, die einer der drei Tags besitzt, spielt einen VoiceClip ab.
Mit dem Dictionary wird versichert, dass der Button auch nur einmal die VoiceClip abspielt, damit bei erneuter Tätigung des Buttons nicht der nächste Clip im nächsten voiceIndex abspielt.

ButtonPlate
Beim Hovern mit der VR Hand wird der Knopfsound abgespielt und die jeweiligen SpawnManager Methode im Event Inspektor aufgerufen.

Can
Wenn die Dose mit etwas collided, spielt es einen Sound ab.

Controller
Hier wird die Logik des Input Managers behandelt.

Dart
Wenn das Dart mit der Dartscheibe collided, dann steckt das Dart an der Scheibe fest.
Für die Task im Schiff wird überprüft, ob der richtige Gravitationsmodus mit der richtigen Kondition (Anfang, zurück von Jupiter / Mond / Erde) eingestellt ist. Solange der Slider in dem Gravitationsmodus ist, wird die Task für den Dart erfüllt. Dasselbe gilt für den Tennisball. Gleichfalls wird 1 / 3 Tasks als abgeschlossen markiert.

Delete
Wenn ein Object mit dieser Collision entern sollte, wird dieses Object gelöscht für Performance.

GameManager
Der GameManager behandelt alle States des Projektes.
Es besitzt ein enum GameState mit drei States: Man fängt bei TaskZero an, TaskOne ist das Dribbeln und TaskTwo ist das Korbwerfen.
Es wird als static instanziiert da es von jeder Szene aufgerufen wird.

GravityManager
Hier wird die ganze Gravitation gespeichert und gesettet. Im Inspektor wird der Gravity Modus an den jeweiligen Scenes eingestellt.

LevelManager
Hier wird der buildIndex gesettet, um zum richtigen Planeten zu kommen. Beim Level laden werden alle Variablen zurückgesetzt. Die Voice wird beim am Anfang der nächsten Szene erst abgespielt.

SpawnManager
Hier wird Basketball, Tennisball, Dose, Dartpfeile und Teleporter an deren jeweiligen Spawnpoints und Spawnrotations instanziiert. Alles, bis auf den Teleporter, wird mit einem Cooldown von einer Sekunde bei Knopfdruck gespawnt. Der Teleporter darf nur einmal in der Szene gespawnt werden.

Teleporter
Wenn man auf den Teleporter hovert, dann wird das nächste Level für den jeweilige Planeten geladen.

Tennisball
Dieselbe Logic wie beim Dartskript, bloß steckt der Tennisball nicht irgendwo bei Aufprall.

TextManager
Hier wird der Gravity Text am Gravity Slider angepasst abhängig vom slider.value und von dem GravityManager enum GravitationalForceMode.
Der Text des Scores für das Dribbeln wird aktualisiert.

UIManager
Der UIManager besitzt die komplette Logic bezüglich der UI.
Der Slider ändert zugleich den Gravitationsmodus und den Text am Monitor.
Je nach Kondition und Szene werden bestimmte Panels und Buttons aktiviert bzw. deaktiviert. Die Simulation soll ein linearer Verlauf sein, weshalb es die bool Checks von isVisitable existiert (Um den Mond besuchen zu können, muss man erst den Jupiter besucht haben etc.).

CreateWithVR Scripts
Beide Scripte OnButtonPress und ToggleRay wurden aus dem CreateWithVR Kurs entnommen und angewendet, um die UI Interaktion mittels Ray Interactor durchführen zu können.

Sound
Alle Sounds sind als .ogg exportiert worden. Alle UI Sounds und Voice Lines wurden auf Mono angepasst, der Rest bleibt Stereo. Dementsprechend wurden alle Audio Sources, abgesehen von UI und Voices, auf 3D einstellt, um die Immersion zu verstärken.
Für alle 5 Szenen gibt es eine einzigartige Musik und Ambience.

Playthrough
1. Man spawnt auf dem Raumschiff, wo die Controls angezeigt wird
2. Beim weiterklicken kommt das Panel von Jupiter mit den ganzen Statistiken
3. Auf dem Task Knopf muss man zunächst die Aufgaben erfüllen, damit man Jupiter besuchen kann
4. Wenn man auf Visit klickt, erscheint ein Portal und man wird zum jeweiligen Planet teleportiert
5. Als erste Aufgabe muss man 3x hintereinander dribbeln
6. Zum Schluss muss man den Basketball in den Korb reinwerfen, damit man zurück ins Schiff gelangt
7. Man macht die Task auf dem Spaceship für den Mond
8. Danach gibt es eine Einführung zum Mond mit denselben Aufgaben und wieder den Planeten besuchen
9. Wiederholung mit Aufgaben auf dem Schiff
10. Dieselben Tasks auf der Erde
11. Am Ende hat man die Möglichkeit, die Planeten wieder besuchen zu können oder zurück nach Hause zu teleportieren

Build testing
Beim Build hat alles geklappt außer eine Sache. Die Emmisionslichter am Korb hat sehr stark geflackert. Die Lösung dazu war, dass ich die Graphics API von OpenGL auf Vulkan gewechselt habe.

Funktionen, die geplant waren
- Der next Button ist ausgeblendet und wird erst angezeigt, wenn der VoiceClip zu Ende ist
- Die Schiff Task ist direkt beim Laden aktiv, nicht, wenn man erst auf Task drückt
- Es sollte ein Failsafe für das Dribbeln existieren, um die Schwierigkeit auf dem Jupiter simpler zu machen
- Die letzte Szene mit Credits fehlt

Existierende Bugs
Manchmal überspringt die VoiceLine einen Index und spielt ein VoiceClip zu früh an. Der Grund ist unbekannt. Einmal ist es passiert, als man von Jupiter zurück zum Schiff kehrt und man danach mit der UI interagiert.
Wenn der Dart die Schreibe trifft und man diesen Dart aufhebt, ist isKinematik noch auf on und bleibt an der Stelle stehen, wo auch immer man das Dart loslässt.
Der Home Teleport bringt den Player zurück zum Raumschiff.

Aufgaben der Teammitglieder
Erwin: Controls, Locomotion, Scripting, Physics Material, Playtesting, Sound Designer, Light Probe, Collisions, Anchors, Interactions, Readme

Thanh: Blockout, Themenrecherche, Level Design, User Interface, Partikelsystem, AI-Stimme, Post Processing, Itch.io Page

Max: 3D Asset Modelling, Texturing, Materials, UV Mapping, Textures, Light Baking, Reflection Probe, Performance Anpassung

Fazit
Die Simulation verschiedener Gravitationen mittels einer VR Applikation hat sehr lange gedauert, um dies so realitätsnah wie möglich zu implementieren, jedoch merkt man schon sehr stark die Unterschiede.

Ich habe einiges über Enums, Dictionaries und sehr viel über VR Anwendungen gelernt.
Für die Manager Skripte werde ich ab sofort Singletons verwenden, um lange Skripte wie die Methode StartingUI im UIManager in der Zukunft zu vermeiden.