**Design Rapport**

**Konceptbeskrivelse**

Vores spil er et simpel 3d parkour spil, der er baseret på minecraft parkour, altså uden alt for mange koncepter, med boost, en speciel gravity og special effects. Der er simple styre egenskaber med tastatur tasterne, WASD, og med musen som et orienterings “kamera” set fra ens eget perspektiv. Der er lige nu to niveauer, et let og et svært, og måden man kommer op i niveau er ved at klare banen, og derefter ramme portalen. Der er to flader i første niveau, der ved berøring lyser en Philips Hue lampe op i Design lokalet på Slotshaven Gymnasium.

Det er lavet så hvis du falder ned, så bliver du sat tilbage til et spawn point, der er foran banen. Første bane er lavet med forholdsvis store flader, og lette udfordringer, der let kan overkommes, hvor den anden bane er lavet sværere i form af mindre flader og rotation i banen.

Vores ideer for spillet er også at vi ville, lave et point system med coins rundt omkring på parkour banen, så du kunne indsamle dem og bruge dem til at købe fx skins, specielle baner eller upgrades. Vi mangler også noget på vores andet niveau, da vores spawn point er dumt, og hvis du falder ud af banen bliver du spawnet ude i ingenting og bliver ved med det. Vores menu mangler også en funktion vi ville tilføje, nemlig at stoppe spillet, og starte forfra.

**Spillet i MDA-termer**

Vores spil er et klart challenge spil, hvor vi har to parkourbaner. Spillet er rimelig ligetil, man starter spillet foran den første parkourbane og gennem WASD-tasterne, samt mellemrum-knappen (Space) kommer igennem parkourbanen. Som andet input end tastatur, kan man også kigge rundt med musen. Gennemføres den første parkourbane, skal man ramme “portal” objektet, der gør at man kommer videre til næste bane (scene).

Spillet er også et sensation spil. Idet man gennemfører en parkourbane, giver det en lettelse at man er kommet igennem sin bane. For at give spillet mere sensation, kunne man havde tilføjet firkanter, som man kunne nå og derefter få point.

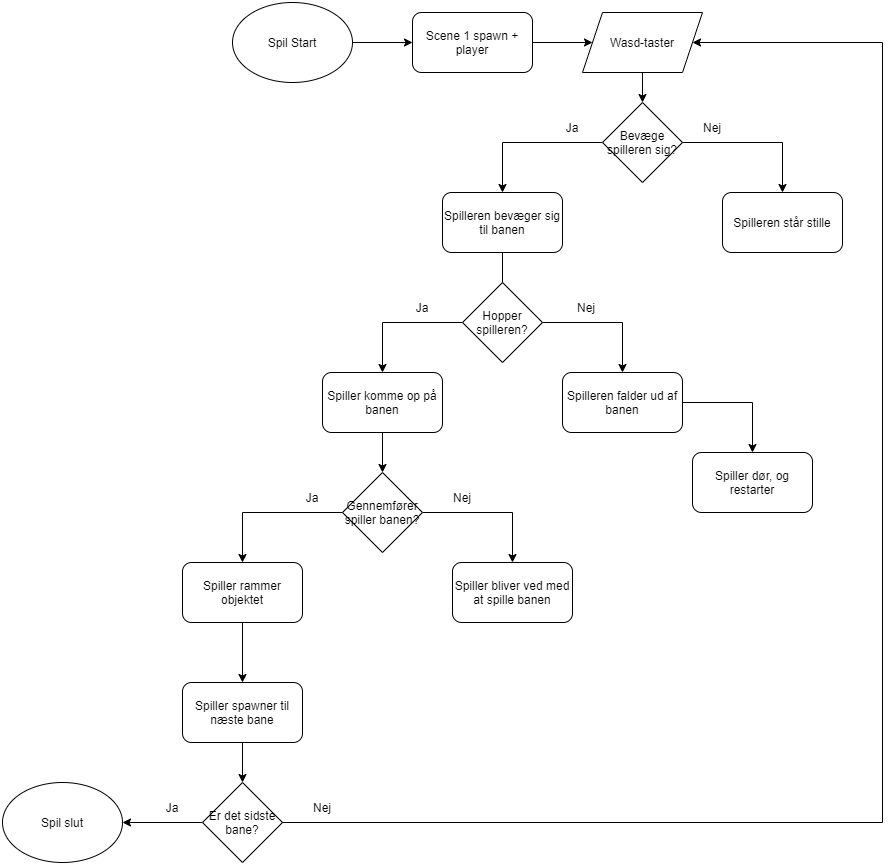
De mechanics der er i spillet er nævnt, men at man kan bevæge sig frem, tilbage, fra side til side, man kan hoppe og man kan kigge rundt med sin mus.

Når spillet kører så er dynamics i spil. Da spillet bliver set som et challenge spil, er det på grund af selve banen samt de flere baner, gør det til en udfordring. Hvis der var implementeret point, kunne det havde øger challenge oplevelsen, da man måske skulle udfordre sig selv endnu mere ved at opnå disse point og dermed også øge sensation.

Vores målgruppe er nok til 12+ år, da det sværhedsgraden godt kan være relativ høj. Hvis man ikke er bekendt med de klassiske spille mechanics som vi har brugt, kan det godt blive svært. Kuglen har nogle dynamics der gør at det godt kan føles at man ikke har styr på den. På grund af den sværhedsgrad, ville det nok være bedst hvis bekendte spillere spille spillet.

**Flowchart/Blokdiagram**

**Flowchart over spil**

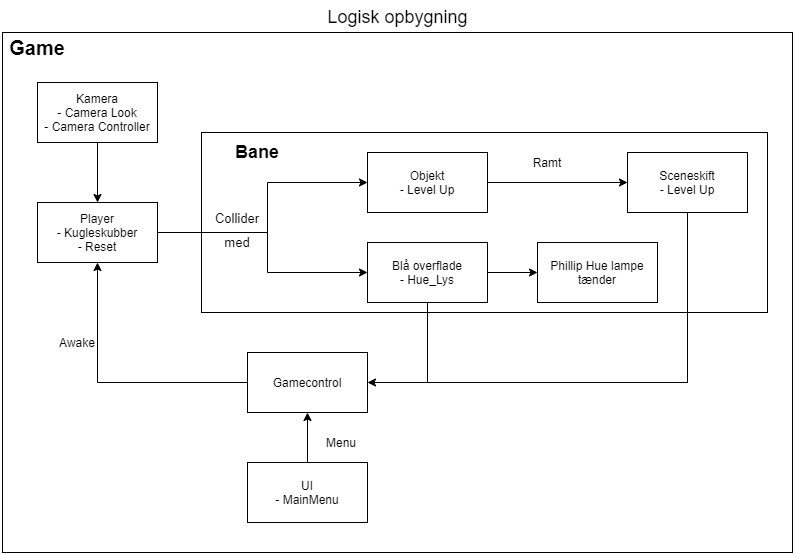


* Spillet starter, hvor hele første scene spawnes. For at spilleren kan bevæge sig, skal spilleren bruge wasd-tasterne, hvis spilleren ikke gør det, kan spilleren ikke rigtig gøre så meget. Ved at komme op på selve parkourbanen, skal spilleren bruge mellemrums-knappen, hvis spilleren ikke gør det, kan de heller ikke gøre særlig meget. Hvis de bare kører derudaf, falder de ud af banen og restarter.

Når Spilleren gennemfører banen, kommer de over på en platform, hvor de kan komme videre til næste bane, med højere sværhedsgrad end den før. Ideen bag spillet er at der mere end to baner, så hvis det er den sidste bane, køres koden en gang mere og spillet sluttes.

* Grunden til der kun er et flowchart er at begge baner har samme princip, det eneste der er anderledes, er opsætningen af spillet og den øget sværhedsgrad.

**Blokdiagram**



* Dette blokdiagram viser den generelle struktur for de overordnede objekter i spillet. Der vises hvilke objekter der kommunikere med hinanden, når der sker en aktion. GamControl gør så at spilleren med kamera banen sættes op. Kameraet er sat til Player gennem de scripts der er sat på, samt i unity. Objektet, eller portal,  er at arbejde sig hen til et objekt der gør at man kommer videre til næste scene, når Player og portal, rammer hinanden skiftet der scene. Derefter kommer GameControl ind igen og starter næste bane op.

Inden man begynder at spille spillet, er der en menu som vises første

**Lister over assets**

Scene 1

Et billede, der indeholder tekst

Automatisk genereret beskrivelse

* MenuCanvas
* Kugle
* Bane
  + Downloaded Asset - PyroParticles/Demo
* End Game objekt
* Kamera

Scripts:

* CameraLook
* GameController
* Hue\_Lys
* Kugeskubber
* LevelUp
* MainMenu
* Reset
* CameraController

Scene 2Et billede, der indeholder tekst, computer, æske

Automatisk genereret beskrivelse

* Bane
* Downloaded Asset - PyroParticles/Demo
* Kamera
* Kugle

Scripts:

* CameraLook
* GameController
* Kugeskubber
* Reset
* CameraController

**Beskrivelse af asset**

Vi har valgt at gå i dybden med Scriptet Kugleskubber.

Et billede, der indeholder tekst

Automatisk genereret beskrivelse

Vores Awake function initializer vores variabler vi bruger til at ændre vores boldt/“player”s adfærd, for eksempel at hoppe eller at bevæge sig frit rundt. Awake funktionen bliver kun kaldt en gang i løbet af instansen af Kugleskubber scriptet.

Et billede, der indeholder tekst

Automatisk genereret beskrivelse

Der var et problem med den hastigheds funktion vi fik fra det originale script, fordi at den opbyggede hastigheden over tid, og det virkede rigtig underligt, og var meget upræcist, så vi skrev koden om til at have en fikset hastighed, der dog plads til forbedringer når den skal stoppe, da den også skal bruge en pæn lang op bremse periode. Koden virker sådan set bare ved at tilføje en kraft i den retning du trykker.

Den læser hvilken tast du trykker i et script der hedder SceneInputs, der er forbundet med det nye input system, og tilføjer så den fikset Force (moveSpeed) til skub i enten y eller x retning.

Så til Jump funktionen, den var utrolig vigtig i vores parkour spil, fordi uden den ville det ikke være parkour. Her tjekker vi om at vores bold rammer en collider, fordi så er den jo ikke på jorden… Ikke helt, men den basale essens er der, hvis den er i kontakt med en collider er isGrounded true, og så udføre den koden, der påvirker vores bolde med en kraft ned af, der så skyder vores bold op af.

Så bliver isGrounded så false, så du ikke kan blive ved med at hoppe, og så bliver bolden påvirket af den gravity der er i spillet, og bliver trukket ned mod jorden.

Et billede, der indeholder tekst

Automatisk genereret beskrivelse

Som forklaret ovenfor tjekker OnCollisionStay om bolden er i kontakt med en collider, og hvis den er, bliver udsagnet isGrounded true, og så kan du hoppe igen.

I vores Update funktion kalder vi Move funktionen, fordi den har brug for at bliver opdateret hver frame for at den bevæger sig “smooth” rundt og ikke lagger, og det er så det update funktionen gør.

**Selvevaluering**

Forløbet har fungeret godt. Der var relativ god kommunikation mellem begge parter, samt uddelegeringen af arbejdet har virket lige. Rose stod for opsætning af banen og Oliver stod for koblingen mellem forskellige objekter/scripts/game mechanics.

Hvis vi havde sat os ned i starten af spillet og havde været mere konkret med hvad spillet skulle indeholde, havde nok sat tid af til at tilføje point.

Vi kunne havde brugt trello noget mere gennem projektet, dog var det en god måde at opstarte projektet, så man kunne få skrevet ned hvad man skulle få lavet. Dog vidste vi ikke helt hvad det var.