Dania
Academy of Higher Education

Academy Profession (AP) Degree in Computer Science

TITLE:

Breakout in Unreal Engine 4

PROJECT PERIOD:
GrnXXdatXX,
December 2015

PROJECT GROUP:
XX

STUDENTS:
Nichlas Bruun
Mathias Forsberg
Bjarne Kristensen

SUPERVISOR:
Jonathan

COPIES: 1

REPORT PAGES: 21

APPENDIX PAGES: 1

TOTAL PAGES: 22

ABSTRACT:

Problemformuleringsspørgsmålet?

Hvorfor dette emne er spændende.

Hvad vi har lavet og fundet ud af.

Indhold

Kapitel 1

Forord

Kapitel 2

Problemformulering

2.1 Problemformulering

Hvordan udvikler man et breakout type spil i Unreal Engine 4?

- Hvilke analytiske værktøjer vil være værd at benytte?
- Hvilke systemudviklingsværktøjer er gode at bruge i sådan et projektforløb?
- Hvilke Unreal Engine 4 værktøjer kan optimere udviklingen?

2.2 Afgrænsning

Da projektets varighed er 4 uger, vil et fuldt testet og balanceret spil nok ikke kunne nås at lave, samtidigt med der skal skrives en systemudviklingsrapport. Der er fokus på koden i spillet, så grafikken er ikke nødvendigvis noget der vil blive brugt tid på, lyddessign er heller ikke en prioritet. Der vil heller ikke være tid til markedsanalyse, dette er fravalgt grundet produkt og systemudvikling er i fokus. Undervisningen har bestået af en uge, og herefter et projektforløb på en uge. Derfor vil noget af tiden også blive benyttet, til at blive familiær med nogle af unreal engine 4's værktøjer.

Kapitel 3

Introduktion

3.1 Unified Process

3.2 Process

Udviklingsforløbet er der planlagt at benytte unified process, og eksamens projektet kommer til at illustrerer en enkelt UP-iteration. I denne iteration vil der være fokus på planlægning og kode. Planlægningen vil bestå både af UP-, objektorienteret analyse- og objektorienteret design-værktøjer. Nogle af de værktøjer der vil være taget i brug, er f.eks. GANT-chart til tidsplanlægning og et analyse-diagram. Disse værktøjer og deres brug, vil blive beskrevet i systemudviklingsrapporten. Efter planlægningen og rapportskrivningen vil udviklingsprocsessen træde i kraft, hvor selve spillet vil blive udviklet. Efter udviklingsprocessen vil der være en kort testfase, hvor diverse fejl og mangler vil blive udredt. Til sidst vil udviklingsprocesen samt reflektioner og konklusioner blive ajourført i systemudviklingsrapporten. En kort opsumering i opremsning: Planlægning, rapportskrivning, udvikling, test og til sidst rapportskrivning.

3.3 Versionsstyring

3.4 Objekt Orienteret Analyse og Design

Efter problemformuleringen, afgrænsningen og spil typen; breakout var bestemt, satte udviklingsteamet sig sammen og lavede et analysediagram. Analysediagrammet er et godt værktøj til, at få alle udviklere på samme spor og tankegang omkring projektet. Analysediagrammet lister produktets objekter, og deres relationer til hinanden. På denne måde fik udviklingsholdet snakket sammen om forholdet

imellem de forskellige objekter, hvilket leder til en mere hensigtsmæssig udviling. Da hele udviklings holdet har samme tankegang omkring hvad hvert objekt indebærer. Efter dette blev use cases udviklet, som beskriver hvilke funktioner brugeren af produktet har adgang til. Use case-værktøjet udpensler derfor hvilke funktioner produktet skal indeholde, set fra brugerens synspunkt. Dette giver udviklergruppen en sans for hvilke funktionaliteter har en sammenhæng mellem objekter. For at gøre dette helt klart, udvikles en hændelses- og funktionstabel. Disse værktøjer udmærker sig ved, at klarificere Hvilke objekter i systemet der har fælles funktionalitet. Udover dette bruges værktøjerne også til at specificere funktionstyper, og kompleksiteten af disse. Til sidst i denne fase laves en systemdefinition, med værktøjet "FACTOR"til hjælp. Systemdefinitionen er en kort tekst der beskriver systemets facetter, dette kan bl.a. være systemets ansvar over for brugeren eller teknologien benyttet til at udvikle systemet. Efter brugen af alle disse værktøjer, er systemet til udviling klarlagt og udviklingen kan begyndes.

Kapitel 4

Object Orienteret Analyse

4.1 Systemdefinition

Functionality
-Bevægelse af paddle, Fjernelse af bricks, Scoring, Liv.
Application Domain
-Klassisk arcade spil, i første iteration fokus på kode og funktionalitet.
Condition
-Udvikles til windows, baseret på klassisk arcade spil.
Technology
-Udvikles i Unreal Engine 4, styres med keyboard, ikke krav til kraftig PC.
Objects
-Paddle, ball, Bricks.
Responsibility
-Et virkende spil.

4.1.1 Sytemdefinition

Unreal Breakout er et klassisk arkade spil, som giver spilleren mulighed for at opleve det klassiske gameplay fra Atari Breakout. Som består af at spilleren bevæger en "paddle"i bunden af skærmen, for at forsøge at holde en bold inden for skærmens rammer, samtidigt med at der skal opnås point ved at fjerne "bricks"i toppen af skærmen. Der vil være fokus på spillets kode og funktionalitet i første UP-iteration. Spillet udvikles til windows pc i Unreal Engine 4, og styres med keyboard. Spillet vil kunne køre på en windows-pc købt indenfor de sidste 5 år. Ansvaret over for spilleren vil i første UP-itereation være et meget basalt virkende spil.

Tabel 4.1: My caption

Navn	Kompleksitet	Type
Start Game	simpel	update
Exit Menu	simpel	update
Exit Game	simpel	update
Release Ball	medium	update / compute
Bounce Ball	medium	update / compute
Move Paddle	simpel	update / compute
Break Brick	simpel	update
Change Score	simpel	update / compute
Show Score	simpel	read

4.2 Funktionstabel

Funktions tabel.

4.3 Analyse Diagram

4.3.1 Menu

Menu klassen kommer til at være et "level-blueprint"i Unreal engine. Den skal håndtere indlæsning af menupunkter, i form af knapper og baggrundsgrafik. Menuen kommer til at håndtere alle "OnClick events"som er i menuen, hvilket i dette tilfælde er en "Play-knap"og en "Quit-knap". Som i samme nævnte rækkfølge, kommer til at starte spillet, og lukke spillet.

4.3.2 Camera

Camera håndtere spillets synspunkt, altså hvad spilleren kan se. Kameraet kan vælge ikke at vise spilleren nogle objekter, som der muligvis har været behov for under udviklingen af spillet. Kameraet kan beskrives som en support klasse, da den ikke direkte ændre på nogle elementer, men bare viser eller ikke viser dem. Samme klasse vil være brugt både på kameraet i menuen, og kameraet i selve spillet.

4.3.3 GameWorld

GameWorld vil ligesom Menu-klassen, være et level-blueprint. GameWorld kommer til at være roden til alt spillets logik, og skal bl.a. indlæse alle spillets elementer når spillet startes og når spillet skifter bane. Klassen kommer også til at lave funktionskald, til de andre klasser der er relevant i spillet f.eks. Brick og paddle.

Tabel 4.2: My caption

Events/Klasser	Paddle	Ball	Brick	Menu	Gameworld	Camera	UI
Player clicks on menu obj				X			
Player moves paddle	X				X		
Player releases ball	X	X			X		
Ball collides with object	X	X	X		X		
Score updates		X	X		X		X

4.3.4 UI

UI-klassen vil holde styr på at vise og opdatere brugergrænsefladen i spillet. Dette vil være at vise og opdatere spillerens point og liv.

4.3.5 Brick

Brick-klassen kommer til at ligge på alle de objekter spilleren skal forsøge at ramme med bolden. Klassen vil håndtere at når bolden kollidere med en "brick"vil denne brick forsvinde, og der vil blive lavet de korrekte funktionskald til at opdatere spillerens score, alt efter hvilken type af brick der er tale om.

4.3.6 Paddle

Paddle-klassen kan tolkes som selve spillerens klasse, denne klasse vil håndtere input fra spilleren. Altså at når spilleren trykker på venstre eller højre pil, bevæger paddlen sig i den korrekte retning. Den vil også give de korrekte funktionskald til bolden, alt efter hvilken del af paddlen rammes, og give den korrekte vinkel med til bolden. Når spillet startes eller når spilleren dør, Vil paddlen også kunne bruges til at starte spillet. Dette gøres ved at bolden sidder fast på paddlen, og skydes afsted med mellemrumstasten.

4.3.7 Ball

Ball er bolden i spillet, og klassen vil håndtere bolden, samtidigt med de tidligere nævnte funktionskald. Bolden vil også stå for vind/tab-konditionen i spillet, ved at tjekke om den er udenfor bunden af skærmen, og lade spilleren miste liv hvis den er udenfor.

4.4 Eventtabel

Event tabel.

4.5 Use Cases

Start Spillet:

Scope:
i menuen.

Description:
Spilleren trykker på *play*-knappen, med musen, i menuen for at komme til *gameworlden*.

Preconditions:
Spilleren skal befinde sig i menuen for at use casen kan startes.

Success Guarantee:
Når use casen er opfyldt vil spillet gå fra menuen til *gameworlden*.

Main Success Scenario:
Spilleren befinder sig i menuen og klikker med musen på *play*-knappen og *gameworlden* vises frem på skærmen.

Extensions:
Befinder spilleren sig allerede i *gameworlden* vil det ikke være muligt at trykke på play knappen da den ikke er vises.

Luk Spillet.

Scope:
i menuen.

Description:
Spilleren kan lukke spillet ved at trykke på *exit*-knappen med musen.

Preconditions:
Spilleren skal befinde sig i menuen for at use casen kan startes.

Success Guarantee:
Når use casen er opfyldt vil spillet blive lukket.

Main Success Scenario:

Spilleren befinder sig i menuen og klikker på *exit*-knappen, med musen, hvorefter spillet vil lukke.

Extensions:
Befinder spilleren sig i *gameworlden* vil *exit*-knappen ikke være vist og spilleren kan derfor ikke interagere med den.

Bevægelse af paddle.

Scope:
I *Gameworld*.

Description:
Spilleren kan ved hjælp af højre og venstre piletast bevæge *paddlen* til henholdsvis højre og venstre.

Preconditions:
Spilleren har startet spillet igennem menuen.

Success Guarantee:
Bliver use casen opfyldt korrekt vil *paddlen* blive bevæget til siderne i takt med at spilleren trykker piletasterne ned.

Main Success Scenario:
a. Spilleren trykker højre piletast ned og *paddlen* bevæger sig til højre.
b. Spilleren trykker venstre piletast ned og *paddlen* bevæger sig til venstre.

Skyd.

Scope:
I *Gameworld*.

Description:
Bolden skydes væk fra *paddlen* ved brug af *space*-knappen, dette kan kun gøre når spillet startes eller efter bolden har været uden for spilbanen.

Preconditions:
Spilleren har startet spillet igennem menuen og har ikke skudt endnu, eller bolden

har lige været udenfor banen.

Success Guarantee:
Bliver use casen opfyldt korrekt vil bolden blive skudt væk fra *paddlen*.

Main Success Scenario:
Spilleren har startet spillet og bolden er stadig placeret på *paddlen*, eller bolden har lige været uden for spilbanen. spilleren trykker på *space*-knappen og bolden bliver skudt væk fra *paddlen*.

Extensions:
Bolden er blevet skudt væk fra *paddlen* og har ikke været udenfor spilbanen og kan derfor ikke blive skudt fra *paddlen* igen da den ikke er placeret der.

Gå tilbage til menuen.

Scope:
i *Gameworld*.

Description:
Spilleren går tilbage til menuen fra *gameworlden* ved at trykke på *escape*-knappen.

Preconditions:
Spilleren har startet spillet og befinder sig i *gameworlden*.

Success Guarantee:
Bliver use casen opfyldt korrekt vil spilleren blive vist menuen igen og *gameworlden* vil lukke.

Main Success Scenario:
Spilleren har startet spillet og trykker på *escape*-knappen, herefter vil menuen blive vist og *gameworlden* vil stoppe.

Extensions:
Befinder spilleren sig allerede i menuen vil *escape*-knappen ikke have nogen effekt.

Kapitel 5

Object Orienteret Design

5.1 Gameplay

Unreal Breakout er et spil med et bat[1], en bold[2], en masse brikker[3], tre lukkede sidder, og en åben side. Formålet er at skyde bolden op og ødelægge alle brikkerne og sørge for at bolden ikke ryger ud af banen i bunden hvor spillerens bat og den åbne side er. Når man rammer en brik, går den i stykker eller skifter farve, og så får man point tilføjet sin score. Når spillet starter har man 3 forsøg(bolde) og bolden er låst fast på éns bat i midten. Når man trykker på en dertil bestemt knap skyder man bolden afsted, og med nogle andre knapper bevæger man battet fra side til side for at undgå at bolden ryger ud af spilområdet. Hvis bolden ryger ud af spilområdet, mister man et forsøg(bold) og en ny bold bliver låst fast til batet og klar til at blive skudt afsted på ny. Når man har opbrugt alle forsøg og den sidste bold ryger ud af spilområdet, slutter spillet og man kan se sin score. Får man til gengæld alle brikker fjernet vinder man banen og den samme bane kommer igen uden at ændre på antal forsøg man har tilbage eller nulstille éns score, og bolden bliver igen fastlåst til batet.

5.2 Grafik

Spillets grafik er fundet på *opengameart.org*, som er en hjemmeside hvor folk lægger grafik op til fri afbenyttelse. Dette betyder at det er uden licens men man kan dog donere penge til de brugere der har lagt grafikken tilgængelig på siden.
Breakout er et gammelt spil udviklet af Atari og derfor er der fundet grafik som er tro mod det klassiske spils grafik. Grafikken ligger nogle spritesheets som er kollager

[1]Paddle på engelsk, bliver brugt i vores paddle klasse.
[2]Ball på engelsk, bliver brugt i vores ball klasse.
[3]Bricks på engelsk, bliver brugt i vores bricks klasse.

af billeder som bliver brugt til at skabe de forskellige elementer i spillet. Når spritesheetet er delt op vil de forskellige brikker blive bygget op til et level. Eftersom spilleren rammer brikkerne med bolden vil de skifte farve, farverne indikere hvor mange gange en brik skal rammes for at blive ødelagt. De grafiske elementer er ikke noget der vil blive sat stor fokus på i spillet, da det er spillets gameplay der skal være grundpillen i det.

5.3 Blueprints

Kapitel 6

Implementering

Kapitel 7

Testing

7.1 Black box test

Testningen foregik ved at have en person som ikke var del af udviklingsgruppen, til at teste specifikke elementer fra spillet. Dette blev gjort ved at lave et spørgeskema, personen der testede spillet skulle udfylde under gennemspilningen. Spørgeskemaet bestod af en række situationer/cases som der kunne krydses af hvis de blev opfyldt. Disse situationer er udviklet ved hjælp af use cases fra den objektorienterede analyse, da disse funktioner skal virke i spillet. Efter gennemspilningen har test personen krydset alle scenarierne ud, som virkende. Dog var der en kommentar om at bolden opførte sig fjollet i nogle situationer.

INDSÆT BILLEDE AF SPØRGESKEMA HER!

Kapitel 8

Reflektion

Kapitel 9

Konklusion