**Zombie Apocalypse Developer Documentation**

Glory Inc

*Processen*

Projektet har byggts runt sprintar som håller en vecka styck. Detta har gjort det möjligt att få in så mycket ‘features’ som möjligt och fortfarande ha ett fungerande projekt.

Veckan har alltid börjat med att gå igenom föregående veckas sprint och kolla vad som inte blev helt klart samt vilka problem som uppstod.. Efter detta gick vi igenom sprintbackloggen och delade upp ett antal av de viktigaste uppgifterna som var kvar.

Sedan jobbade vi både separat och i grupper med uppgifterna. En del av uppgifterna tilldelades flera personer och då fick de antingen dela upp det eller jobba i form av pair programming.

*Komponenter*

Det finns X större komponenter i projektet. Följande listan visar de olika komponenterna där det viktigare är högre upp.

● Spelplan

● Spelaren i form av styrning

● Pathfinding för AI’n

● Uppritning av kartan

● Hantering av vapen och affären

● Animationer

*Design val*

Ett par olika val har gjorts för att hålla en bra struktur i projektet. Bland annat handlar det om hantering av git där vi valt att använda oss av git flow då vi arbetar med olika uppgifter i olika sub-branches för att inte blanda ihop och förstöra varandras arbeten.

När det handlar om projektet i sig har det byggts upp baserat på en Model-View-Controller (MVC) struktur. Detta gör det möjligt för applikationen att utnyttja en effektiv struktur vilket också gör det lättare att implementera de olika delarna. Model hanterar datan. View renderar datan som finns i model för att få fram användargränssnitt och interaktionsmöjligheter. Controller tar sedan hand om användarinteraktioner för att hantera ändringar i model och view.

*UML*

Kommer senare

*Utomstående beroenden*

Applikationen använder sig av grafikbiblioteket libGDX för att kunna stödja de grafiska momenten spelet behöver erbjuda. LibGDX erbjuder nämligen ett API som stödjer grafiska applikationer för bland annat spel av den typ som vår applikation behöver.

Detta API gör det även möjligt att porta koden till både desktop och android utan svårigheter vilket gör det möjligt att med lätthet testa applikationen i desktop miljö istället för att hela tiden koppla upp det till en telefon eller använda en seg android simulator.

Vi använder oss av ett externt fysikbibliotek som heter Box2D. Det är ursprungligen gjort för C++ miljö men Libgdx har portat biblioteket så det är användbart med Java. All spel-logik hanteras av Box2D. Det är uppbyggt så att alla föremål som rör sig runt är av typen Dynamic Body. Samt de föremål som är fast i världen, till exempel hus och parkbänkar, är av typen Static Body. Box2D hanterar kollisionshantering mellan kroppar av sig självt. Spelaren är av typen Dynamic Body och när en spelar försöker gå in I en vägg som är Static så stannar spelaren. Box2D ger oss också ett event vid varje kollision så ifall någon varelse i världen träffas av ett skott så kan vi programmeringsmässigt se detta och hantera det.

Alla kroppar som skapas med hjälp av Box2D kan vi ge olika attribut så att de interagerar annorlunda med varandra. Ett exempel på ett sådant attribut är densitet. Ifall två kroppar med olika densitet kolliderar med varandra så flyger den enda iväg.

*Annan relevant information*