LINKÖPINGS UNIVERSITET

TDP005 Projekt: Objektorienterat system

Designspecifikation

Författare

Agnes Hallberg, agnha531@student.liu.se
Eric Jönsson, erijo137@student.liu.se

INSTITUTIONEN FÖR DATAVETENSKAP

Höstterminen 2018
Version 1.0

2018-11-27

Revisionshistorik

Ver.	Revisionsbeskrivning	Datum
1.0	Första utkast	18-11-27

1 Detaljbeskrivning av Player

Playerklassen ska representera karaktären spelaren kontrollerar. Spelaren har tre olika attacker att välja mellan samt tre riktningar att röra sig i. Spelkaraktären kan inte lämna spelplanen. Klassen Player ärver från överklassen Character. Följande klassmedlemmar finns i Playerklassen:

- int hp; ärvs från Character. Denna datamedlem innehåller information om hur mycket skada spelkaraktären kan ta innan förlust.
- vector<double> position; ärvs från Character. position sparar spelkaraktärens koordinater på banan.
- vector <double > hitbox; ärvs från Character. Datamedlemmen sparar spelkaraktärens storlek.
- double movespeed; ärvs från Character. Dikterar hur snabbt spelkaraktären kan röra sig per tick.
- void move(); ärvs från Character. Medlemsfunktionen kallas genom tangenttryckning av spelaren och uppdaterar spelkaraktärens position. Om spelarens position kommer hamna utanför spelplanens gränser efter en move kommer den inte att genomföras.
- void take_damage(); ärvs från Character. Funktionen minskar spelkaraktärens liv och ger immunitet mot ytterligare skada i ett fåtal sekunder.
- void attack_weak(bool high). En attackfunktion som kallas genom tangenttryckning och försätter spelkaraktären i attackläge. Vid attackläge får spelkaraktären en extra hitbox framför sig som fienden kan ta skada av vid kollision. Det finns två olika attacklägen; ett högt och ett lågt. Vilket som sker beror på vilken tangent spelaren tryckt på.
- void attack_strong(). En attackfunktion som kallas genom tangenttryckning. Försätter spelkaraktären i ett attackläge som fungerar snarlikt det tidigare nämnda attackläget. Denna attack varar dock längre än de alternativa attackerna. Vissa fiender tar enbart skada av detta attackläge.
- void jump(). Spelkaraktären rör sig i y-led till en viss y-koordinat och sedan tillbaka i motsatt riktning.

2 Beskrivning av Enemy

Enemyklassen ska reprensentera fienderna som spelaren behöver besegra. Enemyklassen har tre underklasser; Peasant, Knight samt Village fool. Enemy är en abstrakt klass. Spelkaraktären tar skada när den kolliderar med någon av Enemyobjekten. Fienderobjektens mål är att röra sig in i spelkaraktären. Enemyklassen ärver från Characterklassen.

- int hp; ärvs från Character. Denna datamedlem innehåller information om hur mycket skada Enemyobjektet kan ta innan destruktion.
- vector<double> position; ärvs från Character. position sparar Enemyobjektets koordinater på banan.
- vector <double > hitbox; ärvs från Character. Datamedlemmen sparar Enemyobjektets storlek.
- double movespeed; ärvs från Character. Dikterar hur snabbt Enenmyobjektet kan röra sig per tick.
- void move(); ärvs från Character. Funktionen kallas varje tick och uppdaterar Enemyobjektets position närmre spelkaraktären.

- void take_damage(); ärvs från Character. Objektets hp minskar med 1. Om objektets hp blir noll spelas en kort destruktionsanimation upp och objektet förstörs.
- int immunity. Används i resolve_hit() för att avgöra om Enemyobjektet ska ta skada eller inte.
- void resolve_hit(). Denna funktion kallas när spelkaraktären träffar ett fiendeobjekt med en attack. Om attacken inte motsvarar värdet i immunity tar fienden skada.

2.1 Peasant

Peasant är den vanligaste förekommande fiendetypen och finns i två varianter. Det som skiljer varianterna åt är värdet i immunity. Texturen på fienden visar vilken immunity den har. Peasant rör sig mot spelkaraktären.

2.2 Knight

Knight ska upplevas svårare att besegra än Peasant. Knight rör sig långsammare än Peasant och spelkaraktären. Knights attack är långsam och försätter den i ett attackläge likt spelkaraktärens. Knight är immun mot svaga attacker. Knight kräver fler attacker för att besegras än andra fiendetyper.

- void initiate_attack(). Denna funktion kallas när spelaren är inom räckhåll för Knights attack. Knight står stilla i attackläge under hela sin attack.

2.3 Village Fool

Village fool rör sig inte mot spelkaraktären utan förflyttar sig i x-led rakt över spelplanen. Det är det enda Enemyobjektet som inte behöver besegras av spelaren för att vinna. Dock måste spelaren besegra Village fool får att få maximalt antal poäng.

- void out_of_bounds(). Funktionen kallas då Village fool passerat spelplanens yttre gräns. Funktionen förstör objektet utan att ge spelaren poäng.

3 Designbeskrivning

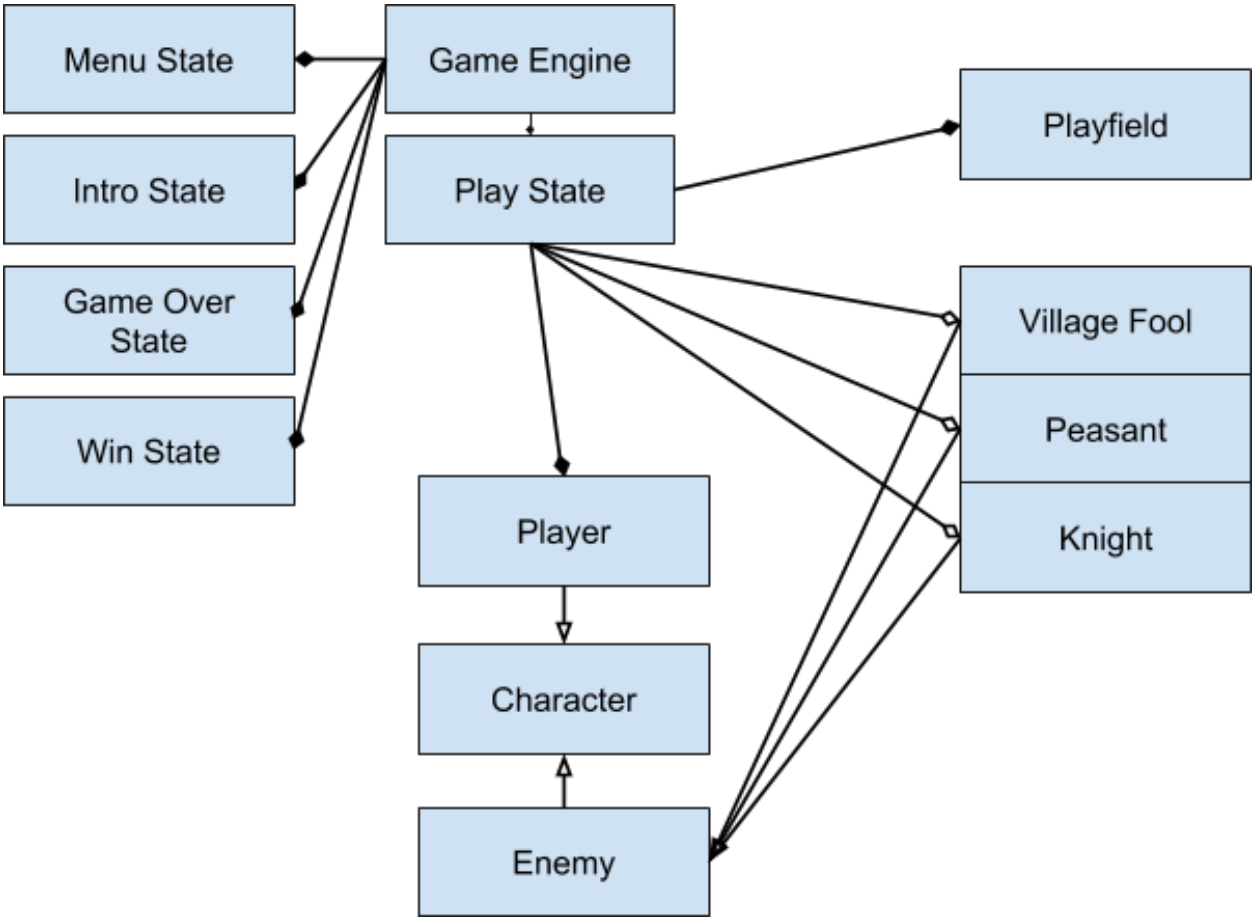
Designen av vårt spel bygger på att vi har en central spelmotor som väljer mellan fem olika spelstadier; PlayState, MenuState, GameOverState, WinState samt IntroState. Vi gjorde detta designval för att enklare kunna expandera spelet med fler gamestates som till exempel cutscenes samt för att kunna komma åt data i olika spelstadier. Designvalet gör utveckling av spelet mer omfattande och om den tillagda funktionaliteten inte utnyttjas till fullo blir det ett slöseri av resurser. En alternativ design hade kunnat vara att skippa spelmotorn och enbart ha två spelstadier att växla mellan. Denna lösning har fler begränsningar men är mindre resurs- och tidskrävande.

Då klasserna Player och Enemy har gemensamma attribut väljer vi att låta dem ärva från samma abstrakta klass, Character. Detta leder till mindre upprepning av kod, vilket i sin tur leder till färre buggar och tydligare kod.

När spelet körs så körs IntroState först och leder sedan till MenuState. Via Menustate kan spelaren sedan komma åt de andra spelstadierna förutom WinState. När PlayState blir anropad läser den in spelplanen från fil. Spelplanen innehåller sitt eget utseende samt hur många fiender som behöver besegras och när de skickas in. PlayState innehåller spelarkaraktären och en vektor av fiender som finns på spelplanen. I varje tick kontrollerar PlayState att ingen fiende befinner sig inom spelkaraktärens hitbox. Om en fiende befinner sig inom spelkaraktärens hitbox kallas take_damage() på spelkaraktären.

Vidare kommer PlayState ansvara för interaktioner mellan objekten på spelplanen samt ansvara för interaktioner mellan objekten och spelplanen.

Följande klassdiagram visar förhållandet mellan klasserna.



4 Externa filer

Spelet hanterar en extern fil för att ladda in spelplanen och för att lagra highscores på den specifika planen. Filen som behandlas ska vara en txt-fil. Den datan som behöver behandlas är spelplanens storlek, fienders startpunkter samt inskickningstider och antal. Vi sparar även en lista med de tre bästa highscores i samma fil så att olika banor får egna highscores..