TDP005 Projekt: Objektorienterat system

Designspecifikation

Författare

Jonathan Dandemar, jonda429@student.liu.se
Jonas Vestlund, jonve547@student.liu.se

Höstterminen 2015
Version 1.0

2015-12-16

Revisionshistorik

Ver.	Revisionsbeskrivning	Datum
1.0	Skapade dokumentet	2015-12-16

1 Systemdesign

2 Detaljbeskrivning av klassen Player

2.1 Generellt

Klassen Player beskriver hur spelarens skepp skall fungera. Player-klassen är unik i avseendet att det är den enda klassen som använder användarinmatad information för att styra sitt beteende. Player-klassen ärver från Moving_Object-klassen som i sin tur ärver från Sprite-klassen. Via arvet från Moving_Object använder sig Player även av AABB-klassen som användsför sin kollisionshantering. Player-klassen genererar även nya instanser av Laser-klassen.

2.2 Konstruktor

Players konstruktor tar emot 8 st parametrar:

1. Ett objekt av Textur-klassen som innehåller dess textur i form av en SDL_Texture-pekare samt höjden och bredden på texturen.
2. Ett startvärde för skeppets x-koordinat.
3. Ett startvärde för skeppets y-koordinat.
4. Skeppets hastighet.
5. Skeppets liv.
6. Skeppets vinkel.
7. En pekare till en vektor som innehåller lasrarna som spelaren skjuter.
8. Ett objekt of Textur-klassen som innehåller lasrarnas textur, höjd och bredd.

Parameter 1-6 skickas som parametrar till Moving_Object-klassens konstruktor. Resterande parametrar används för att initiera medlemsvariabler. Player har även en defaultkonstruktor som sätter de medlemsvariabler som är pekare till nullpointers, samt en kopierings- och tilldelningskonstruktor som båda blivit satta till default.

2.3 Variabler:

Variabel	Ursprung	Syfte
vector<Laser*>* shots_fired	Medlem	En pekare till en vektor innehållandes objekt av Laser-klassen. Det är till den här vektorn nya laserskott läggs till i när spelaren skjuter.
map<string, bool> actions	Medlem	En map som innehåller värden för spelarens input exklusive användandet av supervapnet. Detta omfattar input för rörelse och skjutande av basvapnet.
bool carries_singularity	Medlem	Håller koll på om spelaren har tillgång till supervapnet eller inte.
Texture* laser_texture	Medlem	En pekare till ett objekt av Texture-klassen som innehåller en pekare till Laser-klassens textur samt dess höjd och bredd.
Uint32 shoot_cooldown	Medlem	Håller koll på när spelaren senast avfyrade basvapnet.
AABB this_object	Moving_Object	En variabel av AABB-typ som motsvarar storleken och positionen av spelarens textur och som används för kollisionshantering.
int speed	Moving_Object	Skeppets hastighet.
int current_hp	Moving_Object	Skeppets nuvarande liv.
SDL_Texture* texture	Sprite	En pekare till skeppets textur.
SDL_Rect rect	Sprite	En rektangel som motsvarar skeppets storlek.
double angle	Sprite	Skeppets vinkel.

2.4 Metoder

void update(map<string, bool>)	Medlem	Uppdaterar skeppets position och skapar nya objekt av Laser-klassen om spelaren skjuter.
bool collides(AABB)	Moving_Object	Kollar om skeppet kolliderar med ett annat objekt.
bool is_dead()	Moving_Object	Kollar om spelarens skepp har dött.
void reduce_health(int)	Moving_Object	Sänker skeppets liv.
void draw(SDL_Renderer*)	Sprite	Ritar ut skeppet på spelplanen.

3 Detaljbeskrivning av klassen Enemy_Generator

3.1 Generellt

Klassen Enemy_Generator hanterar skapandet av nya fiender på spelplanen. Klassen ärver ej från någon annan klass, men genererar nya instanser av fiendeklasserna Asteroid, Blaster och Drone.

3.2 Konstruktor

Enemy_Generators konstruktor tar 4 st parametrar:

1. En map innehållande pekare till alla texturer.
2. En pekare till vektorn som lagrar alla fiender.
3. En pekare till vektorn som lagrar alla fienders lasrar.
4. En pekare till ett par som innehåller spelarens nuvarande position.

Konstruktorn initierar samtliga medlemsvariabler, samt lägger till de fyra olika spawnsektionerna (TOP, LEFT, BOTTOM, RIGHT) som används för att generera spawnpunkter till vektorn spawn_sections. Enemy_Generator har även en defaultkonstruktor som initierar alla pekarvariabler till nullpointers, samt en kopierings- och tilldelningskonstruktor som är satta till default.

3.3 Variabler

map<string, Texture*> textures	En map innehållande pekare till alla texturer.
vector<Moving_Object*>* enemies	En pekare till vektorn som lagrar alla fiender.
vector<Laser*>* enemy_shots_fired	En pekare till vektorn som lagrar alla fienders lasrar.
pair<double, double>* player_pos	En pekare till ett par som innehåller spelarens nuvarande position.
vector<pair<char, int» spawn_sections	En vektor innehållandes de olika spawnsektionerna. Består av en char som identifierar positionen samt ett heltal som denoterar intervallet [0, int] som spawnkoordinaten ska slumpa över.
double asteroid_frequency	Fördröjning i sekunder mellan genereringen av nya asteroider.
double blaster_frequency	Fördröjning i sekunder mellan genereringen av nya fiender av typen Blaster.
double drone_frequency	Fördröjning i sekunder mellan genereringen av nya fiender av typen Drones.
Uint32 current_time	Nuvarande tid, används som referenspunkt för tidsjämföring.
Uint32 last_asteroid	Tidpunkten för senast skapad asteroid.
Uint32 last_blaster	Tidpunkten för senast skapad Blaster.
Uint32 last_drone	Tidpunkten för senast skapad Drone.
int asteroid_spawn_count	Antal gånger som nya asteroider skapats.
int blaster_spawn_count	Antal gånger som nya Blasters skapats.
int drone_spawn_count	Antal gånger som nya Drones skapats.

3.4 Metoder

Enemy_Generator har endast en publik metod: **update()**, som uppdaterar antalet fiender om nog lång tid har passerat sen senaste gången detta skett.

4 Diskussion kring design

5 Externa filformat

De enda externa filformat som använts är en .txt-fil för lagring av highscore, samt en .ttf-fil för att importera typsnittet som spelet använder för att skriva ut text.