Functioneel ontwerp T^NKS!

1 APRIL

Klas: ITA 1DD
Gemaakt door: Osama Halabi en Robert Boudewijn
Studenten nummer: 628160 en 631286
Docent: ir. B. van der Wal

HAN_UNIVERSITY OF APPLIED SCIENCES

Inhoudsopgave

1 Inleiding

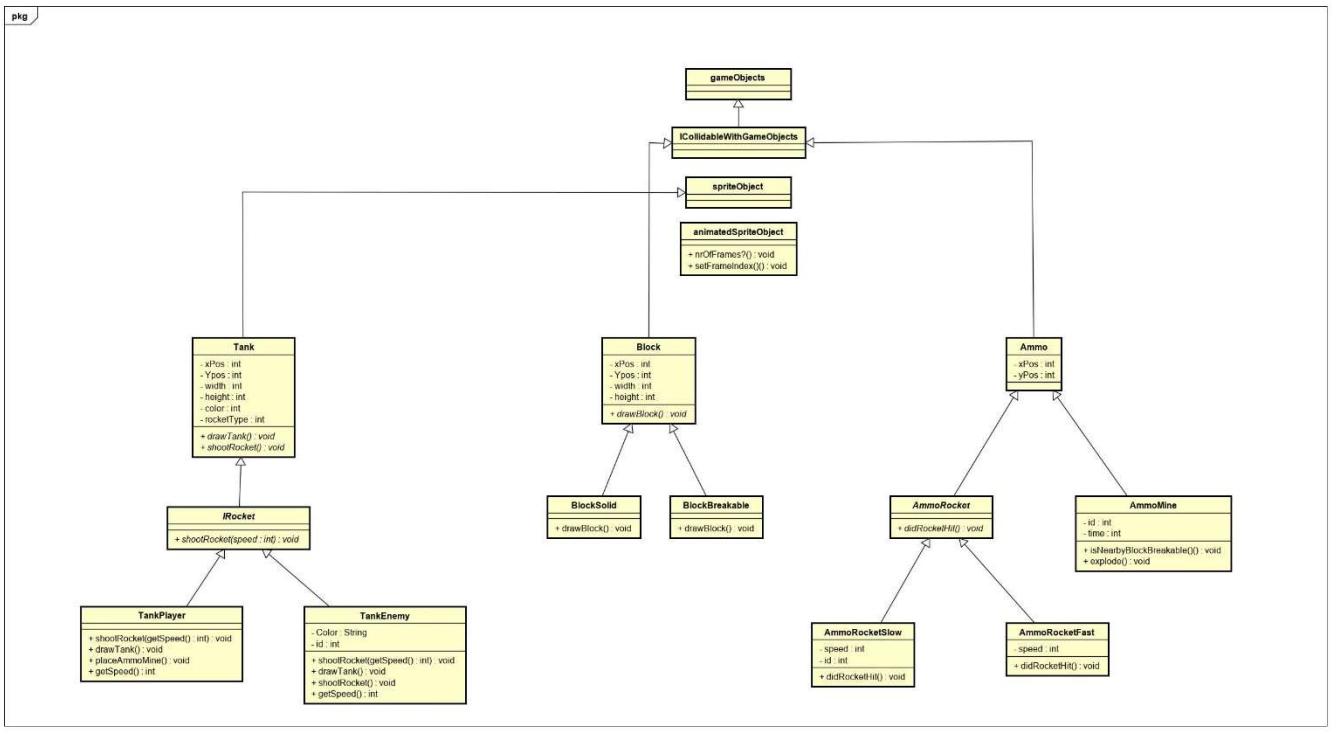
Voor het vak OOPD van onze opleiding moeten wij een spel maken voor afronding van het vak. In dit document behandelen wij alle Technische onderdelen voor het bouwen van dit spel. Hierbij moet u denken aan alle klassen en objecten.

Het spel heet ‘T^NKS!’ en is gebaseerd op het WII Play spel Tanks. (Onbekend, 2019)

Het doel van het spel is om in elk level alle andere tanks uit te schakelen door ze te beschieten met kogels of door mijnen te plaatsen. Het spel speelt zich af in een level ter grote van het speelscherm. Met verschillende obstakels.

2.0 Class Diagram

Voor het bouwen van dit spel hebben wij eerst een klassen diagram moeten uitwerken, deze kunt u hier onder zien.



Figuur 1; Klassen diagram

We hebben we volgende klassen:

2.1 Tank

Met de klas Tank maken we een object van een tank en deze klas overerft van de klas spritObject van de gameEnige .

Deze klas heeft private attributen:
Xpos: int; (hiermee geven we de x positie aan).
Ypos: Int; (hiermee geven we de y positie aan)
Width: Int ; (hiermee geven we de width aan)
Height: Int; (hiermee geven we de height aan);
Color: Int; (hiermee geven we de kleur aan).
RocketType: Int; (hiermee geven we de type aan);

Deze klas heeft public methoden:

- *DrawTank(): Void ;*

- *ShootRocket(): Void;*

2.2 TankPlayer

Deze klas overerft van de klas Tank en implementeert van de abstract klas IRocket.
Deze klas heeft methoden:

- ShootRocket(getSpeed()): void;

Met deze methode overriden we de mehode van de klas IRocket, hiermeer tekenen we de shoot rocket met de gehaalde snelheid van de bepaalde rocket.

- DrawTank(): Void;

Met deze methode override we de methode van de klas Tank en hiermee tekenen we de tank van de player.

- PlaceAmmoMin(): Void;

Met deze methode roepen we een Ammo die wordt van de tank geschoten.

- GetSpeed(): Void;

Met deze methode halen we de snelheid van de gevraagde rocket of slow of fast.

2.2 TankEnemy

Deze klas heeft private attributen:

- Color. Int; (hiermee geven verschillende kleuren van de enmey met deze kleuren bepalen we wat voor snelheid deze tank heeft.
- Id. Int (hiermeer geven we elke Tank een id nummer zodat als een Tank wordt geraakt dat deze tank van het spel weg gaat.

Deze klas heeft de methoden:

- ShootRocket(getSpeed()): void;

Met deze methode overriden we de mehode van de klas IRocket, hiermeer tekenen we de shoot rocket met de gehaalde snelheid van de bepaalde rocket.

- DrawTank() : Void;

Met deze methode override we de methode van de klas Tank en hiermee tekenen we de tank van de player.

PlaceAmmoMin(): Void;

Met deze methode roepen we een Ammo die wordt van de tank geschoten.

GetSpeed(): Void;

Met deze methode halen we de snelheid van de gevraagde rocket of slow of fast.

2.2 block

Block is de parent van alle blokken en erft zelf over van GameObjects. Hij heeft de volgende attributen:

- xPos - int
- yPos – int
- width – int
- height – int

en de volgende methodes:

- drawBlock()

drawBlock() is een methode die overerft naar de child klassen en met deze methode kan je een block tekenen in het speel veld.

2.2.1 BlockSolid

Deze klas overerft van de klas block en heeft maar een methode:

- DrawBlock(): void;

Met deze override we de methode van de klas block en hiermee teken we een solid block.

2.2.2 BlockBreakable

Deze klas overerft van de klas block en heeft maar een methode:

- DrawBlock(): void;

Met deze override we de methode van de klas block en hiermee teken we een breakable block.

2.3 Ammo

De Klasse Ammo is de parent klasse van alle rakketten en mijnen van het spel. En overerft van de game engine de klasse game objects. Omdat deze de parent is erven alle child klassen zijn attributen. Deze zijn:

- xPos – int
- yPos – int
-

2.3.1.1 AmmoRocketSlow

AmmoRocketSlow is een type raket. Dit is eigenlijk een standaard raket die dus ook langzaam vliegt. Hij heeft daarom ook alleen maar het attribuut:

- speed – int

en hij over erft de volgende methode:

- didRocketHid() : void

Deze methode check of een rakket een muur of een andere tank raakt.
Verder over erft hij de attributen van Ammo.

2.3.1.2 AmmoRocketFast

AmmoRocketSlow is een type raket. Dit is een speciale raket die sneller vliegt. Hij heeft daarom ook alleen maar het attribuut:

- speed – int

en hij over erft de volgende methode:

- didRocketHid() : void

Deze methode check of een rakket een muur of een andere tank raakt.

Verder over erft hij de attributen van Ammo.

2.3.2 AmmoMine

AmmoMine is een klasse die een het object mijn maakt. Hij heeft de volgende attributen:

- id : int
- time : int

We willen een attribuut id hebben zodat wij in de gaten kunnen houden hoeveel huidige actieve rakketen we hebben. En time houdt bij wanneer er een mijn is neer geplaatst en wanneer deze moet ontploffen.
AmmoMine heeft de volgende methodes:

- isNearbyBlockBreakable()
- explode()

isNearbyBlockBreakable() kijkt welke blokken de mijn kan gaan laten ontploffen wanneer hij moet ontploffen.
explode() zorgt voor de werkelijke ontploffing. Wanneer de bom moet ontploffen zorgt deze methode er voor dat de blokken of tanks ook kapot gaan.

We hebben de volgende interfaces:

2.4.1 *IRocket*

Deze klas is abstract en heeft maar een methode:

- ShootRocket(int speed) : void;

Met deze interfase kunnen we bepaalen wat is de snelheid van een shot van de tank.

2.4.2 AmmoRocket

Deze klas heeft maar een methode:

- didRocketHid() : void

Deze methode check of een rakket een muur of een andere tank raakt.
Verder over erft hij de attributen van Ammo.

3. Opmerkingen:

Wij willen de enemy tanks anders gaan uitwerken dan normaal. Dit willen we doen door een switch case te gebruiken waardoor we maar 1 klasse hoeven te hebben voor meerdere types enemy’s. Dit maakt het misschien lastiger om bij te houden welk type vijanden we hebben maar daardoor kunnen we wel gemakkelijker uit programmeren.

4. Conclusie

Met deze klas diagram gaan we ons spel maken en we weten nog niet of dat deze klas diagram klopt maar we gaan hem programmeren en we komen daarna uit of we nog iets moeten aanpassen

Figuren

Bronnen

Geen gebruikte bronnen