SPACE RAIDER

TECHNISCH ONTWERP

Versie 1 | Informatica Communicatie Academie | Nijmegen

01 – 04 – 2020

Vak: Object Oriented Program Development

Docent: Herman Telman

Ontwerp geschreven door:

Lauren Meenhorst SN632206 en Yoran Koppes SN638657 Klas 1DC

Table of Contents

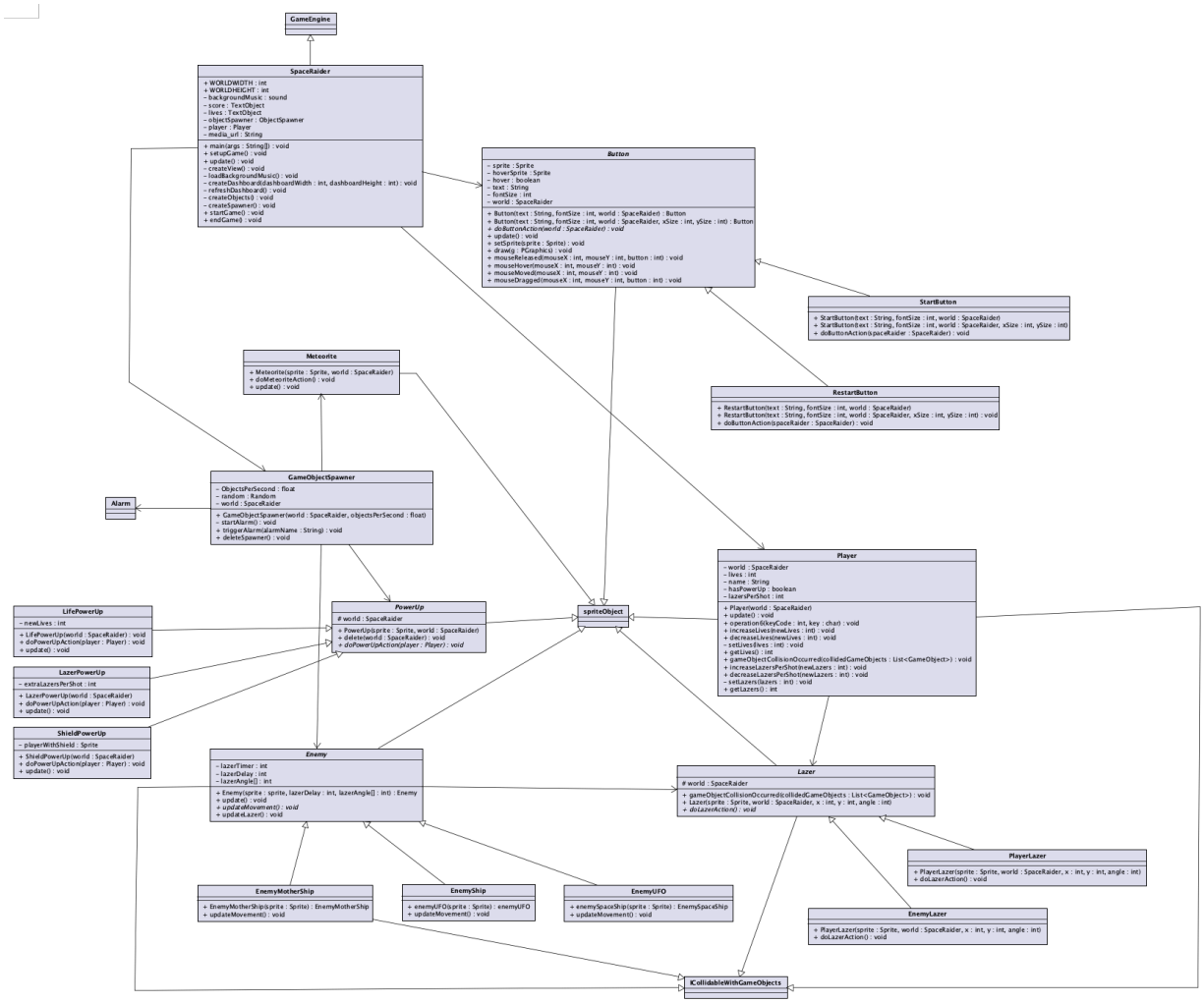
Inleiding

In het Functioneel ontwerp is beschreven wat Space Raider allemaal moet kunnen. In dit document wordt dit uitgewerkt in een Technisch ontwerp.

Het document bevat een Class diagram en een beschrijving van alle klassen.

1. Class diagram

Onderstaand diagram is een grafische vormgeving van alle classes en hun onderlinge relaties.



Figuur 1 Class diagram Space Raider ook te vinden in de map "diagrammen".

2. Software beschrijving

2.1 GameEngine

De GameEngine class representeert de kern van de game.

2.2 SpaceRaider

SpaceRaider is de main class van het programma zij extend GameEnige om gebruik te kunnen maken van de game engine.

Attributen

WORLDWIDTH	int	De breedte van het spel.
WORLDHEIGHT	int	De hoogte van het spel
backgroundMusic	Sound	De achtergrondmuziek die te horen is in het spel.
score	textObject	De score van de speler.
lives	textObject	Het aantal levens van de speler als tekst.
objectSpawner	ObjectSpawner	De spawner om objecten aan te maken in het spel.
player	Player	De speler.
MEDIA_URL	String	Deze string maakt het makkelijk om de media map te bereiken.

Methoden

main(String[] args)	Constructor	De main constructor. Deze is nodig om de game aan te maken.
setupGame()	void	Hierin wordt de initiële view met startknop aangemaakt.
update()	void	
createView()	void	Maakt een scherm aan met een achtergrond image
loadBackgroundMusic()	void	Haalt de muziek uit de media map en loopt deze.
createDashboard(int dashboardWidth, int dashboardHeight)	void	Maakt een scorebord aan dat de score en het aantal levens van de speler toont.
refreshDasboardText()	void	Vernieuwt de tekst die in het scorebord wordt getoond.
createObjects()	void	Maakt de objecten aan die niet worden gespawned. Dit zijn de speler en het scorebord.
createSpawner()	void	Maakt de spawner aan waarna objecten op het scherm zullen verschijnen.
startGame()	void	Wanneer de start knop wordt ingedrukt wordt deze functie aangeroepen. Zij verwijdert alle game objecten en roept hierna createObjects() en createSpawner() aan.
endGame()	void	Wanneer het spel voorbij is wordt deze functie aangeroepen. Zij verwijdert alle game objecten en roept hierna deleteSpawner() en deleteDashboard() aan.

2.3 *Button*

Button is een abstracte class die gebruikt kan worden om knoppen te maken die effect hebben op het spel. In deze class staan functies die de button vormgeven, bepalen of er een klik heeft plaatsgevonden en acties uit voeren.

Attributen

sprite	Sprite	De standaard afbeelding die wordt meegegeven wanneer er een button wordt aangemaakt.
hoverSprite	Sprite	De afbeelding die getoond wordt wanneer een cursor zich op de button bevindt.
text	String	De tekst die over de button wordt geplaatst.
fontSize	int	Specificeert de grootte van text.
world	SpaceRaider	Nodig om de button in de game te plaatsen.

Methoden

Button(String text, int fontSize, SpaceRaider world)	Constructor	In de constructor worden de class attributen geïnitialiseerd.
Button(String text, int fontSize, SpaceRaider world, int X, int y)	Constructor	In de constructor worden de class attributen geïnitialiseerd.
doButtonAction(SpaceRaider world)	Abstract void	Deze methode wordt overschreven in de button subclasses.

2.3.1 StartButton

In setupGame wordt eenmalig een start button object aangemaakt. Het object start het spel wanneer er op gedrukt wordt. De class heeft geen attributen.

Methoden

StartButton(String text, int fontSize, SpaceRaider world)	Constructor	In de constructor worden de class attributen geïnitialiseerd.
StartButton(String text, int fontSize, SpaceRaider world, int X, int y)	Constructor	In de constructor worden de class attributen geïnitialiseerd.
doButtonAction(SpaceRaider world)	Abstract void	Roept de functie startGame() aan.

2.3.2 RestartButton

Wanneer het spel voorbij is wordt een RestartButton object aangemaakt. Deze geeft de speler de mogelijk opnieuw te spelen. De class heeft geen attributen.

Methoden

StartButton(String text, int fontSize, SpaceRaider world)	Constructor	In de constructor worden de class attributen geïnitialiseerd.
StartButton(String text, int fontSize, SpaceRaider world, int X, int y)	Constructor	In de constructor worden de class attributen geïnitialiseerd.
doButtonAction(SpaceRaider world)	Abstract void	Roept de functie startGame() aan.

2.4 Alarm

De class Alarm maakt deel uit van de game engine. Wij gebruiken deze class om objecten te spawnen. Zij implementeren IAlarmListener om naar een alarm te luisteren en een actie uit te voeren wanneer het triggerAlarm wordt aangeroepen.

2.5 GameObjectSpawner

De class GameObjectSpawner implementeert IAlarmListener om hiermee een bepaald aantal keer per seconde een object te creëren.

Attributen

objectsPerSecond	float	Specificeert hoe veel objecten er per seconde moeten worden gecreëerd.
random	Random	Een willekeurig getal dat gebruikt wordt om te bepalen welk object gecreëerd zal worden.
world	SpaceRaider	Nodig om objecten in de game te plaatsen.

Methoden

GameObjectSpawner(SpaceRaider world, float objectsPerSecond)	constructor	In de constructor worden de class attributen geïnitialiseerd.
startAlarm()	void	In de constructor worden de class attributen geïnitialiseerd.
triggerAlarm(String alarmName)	void	Maakt een array van objecten, kiest dan een random object en geeft deze een random positie. Hierna wordt het alarm gestart als dan de tijd voorbij is die tussen object voorbij dient te gaan wordt het object geplaatst.

2.6 SpriteObject

De SpriteObject class is een subclass van Object. De draw methode is voor deze class niet nodig het is mogelijk een object grafisch te maken met een Sprite. Wij gebruiken de class om onze objecten in de game af te beelden.

2.7 ICollidableWithGameObjects

De interface ICollidableWithGameObjects wordt in ons spel gebruikt om te bepalen of objecten elkaar hebben geraakt. De methode wordt toegevoegd aan object classes die die direct effect merken door collision zoals: Player en Lazer.

2.8 Player

De class Player bevat de methoden die nodig zijn om de speler het spel daadwerkelijk te spelen. Het bewegen van de speler sprite en het aanpassen van zijn spelers spelen hierin de grootste rol.

Attributen

world	SpaceRaider	Nodig om speler in de game te plaatsen.
lives	int	Het aantal levens van de speler.
name	String	De naam van de speler.
hasPowerUp	boolean	Is waar wanneer de speler een powerup in bezit heeft.
score	int	De spelers score.
lazersPerShot	int	Hoeveel lazers er per shot uit player komen.

Methoden

Player(SpaceRaider world)	constructor	In de constructor worden de class attributen geïnitialiseerd.
update()	void	Regelt beweging, bijhouden score en bijhouden levens.
keyPressed(int keyCode, char key)	void	Als toets ingedrukt doe actie.
increaseLives(int newLives)	void	Hoogt het aantal levens van de speler op en roept daarna de functie setLives aan.
decreadeLives(int newLives)	void	Verlaagt het aantal levens van de speler en roept daarna de functie setLives aan.
setLives(int lives)	Void	Maakt lives gelijk aan de lives die worden mee gegeven.
getLives()	void	Retourneert het aantal levens van de speler.
gameObjectCollisionOccurred(List<GameObject> collidedGameObjects)	void	Als de speler een object raakt doe een actie afhankelijk van het object type wat geraakt is.
increaseLazersPerShot(int newLazers)	void	Hoogt het aantal lazers van de speler op en roept daarna de functie setLazers aan.
decreaseLazersPerShot(int newLazers)	void	Verlaagt het aantal levens van de speler en roept daarna de functie setLazers aan.
setLazers(int lazers)	void	Maakt lazersPerShot gelijk aan de lazers die worden meegegeven.
getLazers()	void	Retourneert het aantal lazersPerShot van de speler.

2.9 Meteorite

De class meteorite wordt gebruikt door ObjectSpawner om meteorieten te maken. Ze heeft geen attributen wel methoden.

Methoden

Meteorite (Sprite sprite, SpaceRaider world)	constructor	In de constructor worden de class attributen geïnitialiseerd.
doMeteoriteAction()	void	Er wordt 1 leven van de speler af genomen.
update()	void	Functie die hoort bij SpriteObjects.

2.10 *Enemy*

De abstracte Enemy class is de super class van alle vijandelijke voertuigen

Attributen

lazerTimer	int	Dit is een teller die bijhoudt hoeveel frames het geleden is dat de Enemy heeft geschoten
lazerDelay	int	Dit is het aantal frames dat voorbij moet zijn voordat de enemy weer schiet
lazerAngle	int[]	Dit is een lijst van hoeken die bepaalt in welke richting de lazers moeten vliegen als de Enemy schiet.

Methoden

Enemy(Sprite sprite, int lazerDelay, int lazerAngle[])	void	Deze constructor initialiseert de class atributes en zet een timer.
Update()	void	Deze methode roept updateLazer() en updateMovement() aan.
updateMovement()	void	Deze methode wordt aangeroepen in de update, maar wordt geimplementeerd in de subklassen.
updateLazer()	void	Deze methode telt bij lazerTimer 1 op, daarna checkt hij of de tijd voorbij is. Als dat het geval is “spawnt” hij de lazers.

2.10.1 EnemyShip

Objecten van deze klasse zijn van het eerste type vijand. Het object is een ruimteschip dat 2 lazers schiet per schot.

Methoden

EnemyShip (Sprite sprite)	constructor	Deze methode roept alleen de superconstructor aan, daarbij worden de waarden voor lazers aangeroepen, die zijn specifiek voor EnemyShip. Ze hoeven dus niet als parameters in deze constructor te staan.
updateMovement()	void	Deze methode verandert de y met -1 en bepaalt met een formule op basis van de y een nieuwe x.

2.10.2 EnemyUFO

Objecten van deze klasse zijn van het tweede type vijand. Het object is een UFO die één lazer schiet.

Methoden

EnemyUFO (Sprite sprite)	void	Deze methode roept alleen de superconstructor aan, daarbij worden de waarden voor lazers aangeroepen, die zijn specifiek voor EnemyUFO. Ze hoeven dus niet als parameters in deze constructor te staan.
updateMovement ()	void	Deze methode verandert de y met -1 en bepaalt met een formulet op basis van de y een nieuwe x.

2.10.3 EnemyMotherShip

In de game wordt er slechts 1 EnemyMotherShip object gemaakt. Het is een groot schip dat 5 lazers schiet per schot.

Attributen

nrOfLives	int	houdt bij hoe vaak EnemyMotherShip nog geraakt kan worden voordat hij wordt verslagen.

Methoden

EnemyMotherShip (Sprite sprite)	void	Deze methode roept de superconstructor aan, daarbij worden de waarden voor lazers aangeroepen, die zijn specifiek voor EnemyMotherShip. Ze hoeven dus niet als parameters in deze constructor te staan. Ook wordt de waarde nrOfLives op 3 gezet.
updateMovement ()	void	Deze methode verandert de y met -0.5 en doet niks met de x.
gameObjectCollisionOccurred(List<GameObject> collidedGameObject)	void	Vervangt de collision van Enemy. Verandert nrOfLifes met -1 en kijkt of er nog 1 of meer over zijn. Als dat niet het geval is verwijdert hij zichzelf en brengt het spel naar het eindscherm.

2.11 *PowerUp*

De abstracte Enemy class is de super class van alle powerups.

Artributen

world	Spaceraider	Nodig om powerups te plaatsen.

Methoden

PowerUp(Sprite sprite, SpaceRaider world)	constructor	Maakt een SpriteObject aan in world.
delete(SpaceRaider world)	void	Verwijdert de powerup.
doPowerUpAction(Player player)	abstract void	Voert de actie uit van de powerup. Deze methode wordt in de powerUp subclasses overschreven.

2.11.1 LifePowerUp

Objecten van deze klasse zijn het eerste type powerup, als de speler deze oppakt krijgt hij er een leven bij.

Attributen

newLives	int	Bepaalt het aantal levens dat de speler er bij krijgt als hij de powerup op pakt.

Methoden

LifePowerUp(Sprite sprite, SpaceRaider world)	constructor	Geeft de superconstructor een sprite mee om een LifePowerUp te maken.
doPowerUpAction(Player player)	abstract void	Roep increaseLives() aan.
update()	void	Doet niets.

2.11.2 LazerPowerUp

Objecten van deze klasse zijn het tweede type powerup, als de speler deze oppakt kan hij tijdelijk meerdere lazers tegelijk schieten.

Attributen

extraLazersPerShot	int	Bepaalt het aantal lazers dat de speler extra schiet als hij de powerup op pakt.

Methoden

LazerPowerUp(Sprite sprite, SpaceRaider world)	constructor	Geeft de superconstructor een sprite mee om een LazerPowerUp te maken.
doPowerUpAction(Player player)	abstract void	Roep increaseLazersPerShot() aan.
update()	void	Doet niets.

2.11.3 ShieldPowerUp

Objecten van deze klasse zijn het derde type powerup, als de speler deze oppakt krijgt een schild.

Attributen

playerWithShield	Sprite	De nieuwe player sprite wanneer hij een shield heeft.

Methoden

ShieldPowerUp(Sprite sprite, SpaceRaider world)	constructor	Geeft de superconstructor een sprite mee om een ShieldPowerUp te maken.
doPowerUpAction(Player player)	abstract void	Maakt de speler tijdelijk onverslaanbaar.
update()	void	Doet niets.

2.12 *Lazer*

Lazers zijn kleine streepjes die de spelers of de vijanden kunnen schieten. Deze klasse heeft geen Attributen.

Methoden

Lazer(SpaceRaider worldint, int damage, int angle, Sprite sprite, int x, int y)	constructor	stelt alle waarden in om de lazer op de goede plek te spawnen en te laten bewegen.

2.12.1 PlayerLazer

Objecten van dit type zijn lazers die de speler heeft geschoten. Deze klasse heeft geen Attributen.

Methoden

void PlayerLazer(SpaceRaider worldint, int damage, int angle, Sprite sprite, int x, int y)	constructor	geeft alle waarden door aan de superconstructor.
gameObjectCollisionOccurred(List<GameObject> collidedGameObjects)	void	zorgt er voor dat de lazer verwijderd wordt als hij iets raakt (behalve powerups en lazers). als hij een Enemy raakt wordt die ook verwijderd.

2.12.2 EnemyLazer

Objecten van dit type zijn lazers die een vijand heeft geschoten. Deze klasse heeft geen Attributen.

Methoden

void EnemyLazer(SpaceRaider worldint, int damage, int angle, Sprite sprite, int x, int y)	constructor	geeft alle waarden door aan de superconstructor.
gameObjectCollisionOccurred(List<GameObject> collidedGameObjects)	void	zorgt er voor dat de lazer verwijderd wordt als hij iets raakt (behalve powerups en lazers). Als hij de speler raakt verliest de speler een leven.