**Systeembeschrijving**

**SE1 KBS1**

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
| Auteurs | Wilco Rook  Youri Dekker  Joery Oost  Ruben van de Kamp  Mika van den Brink  Mick Nieweg | Studiejaar | 2018 / 2019 |
|  | Versie | 1 |
|  | Datum | september 2018 |
|  | Nummer | 1 |
|  |  |  |  |

# Inhoudsopgave

[**Inhoudsopgave**](#_pn2g9tjl9y2b) **2**

[**Inleiding**](#_j07d7074wv2y) **3**

[**Algemeen**](#_dm75cdlnngk) **4**

[**Klassendiagram**](#_x53bha5srld8) **5**

[**Gebruik van events**](#_bmmboaelqb2b) **6**

[**Sequentiediagram**](#_ulsmafc44bwf) **7**

# Inleiding

Om een beter beeld te krijgen over hoe de applicatie technisch in elkaar zit is het belangrijk om te weten waarom we bepaalde keuzes hebben gemaakt en hoe de code in elkaar steekt. Dit hopen wij allemaal voldoende duidelijk te maken in dit verslag. In dit verslag is een klassendiagram beschreven waarin de relaties tussen alle klassen zichtbaar zijn gemaakt. Verder hebben wij een sequentiediagram toegevoegd om te laten zien hoe objecten veranderen. Ook leggen wij uit hoe wij events hebben gebruikt en wat de redenen achter het gebruik zijn.

# Algemeen

De applicatie is gemaakt in de taal C# met het .NET framework van microsoft. Om de applicatie overzichtelijk te houden hebben wij het MV(C) pattern gebruikt, dit houdt in dat we de model, view en controller gescheiden houden. Hierdoor kun je makkelijker de code bijhouden en makkelijker unit testen waardoor het geheel een stuk overzichtelijker wordt.

De MainWindow (waar alles op wordt getoond) bevat zelf bijna geen functionaliteit, behalve dat het bekijkt wanneer de gebruiker een knop indrukt. Alle functionaliteit wordt eigenlijk pas aangemaakt en toegepast in de klasse game. Deze klasse wordt wel aangeroepen door MainWindow.

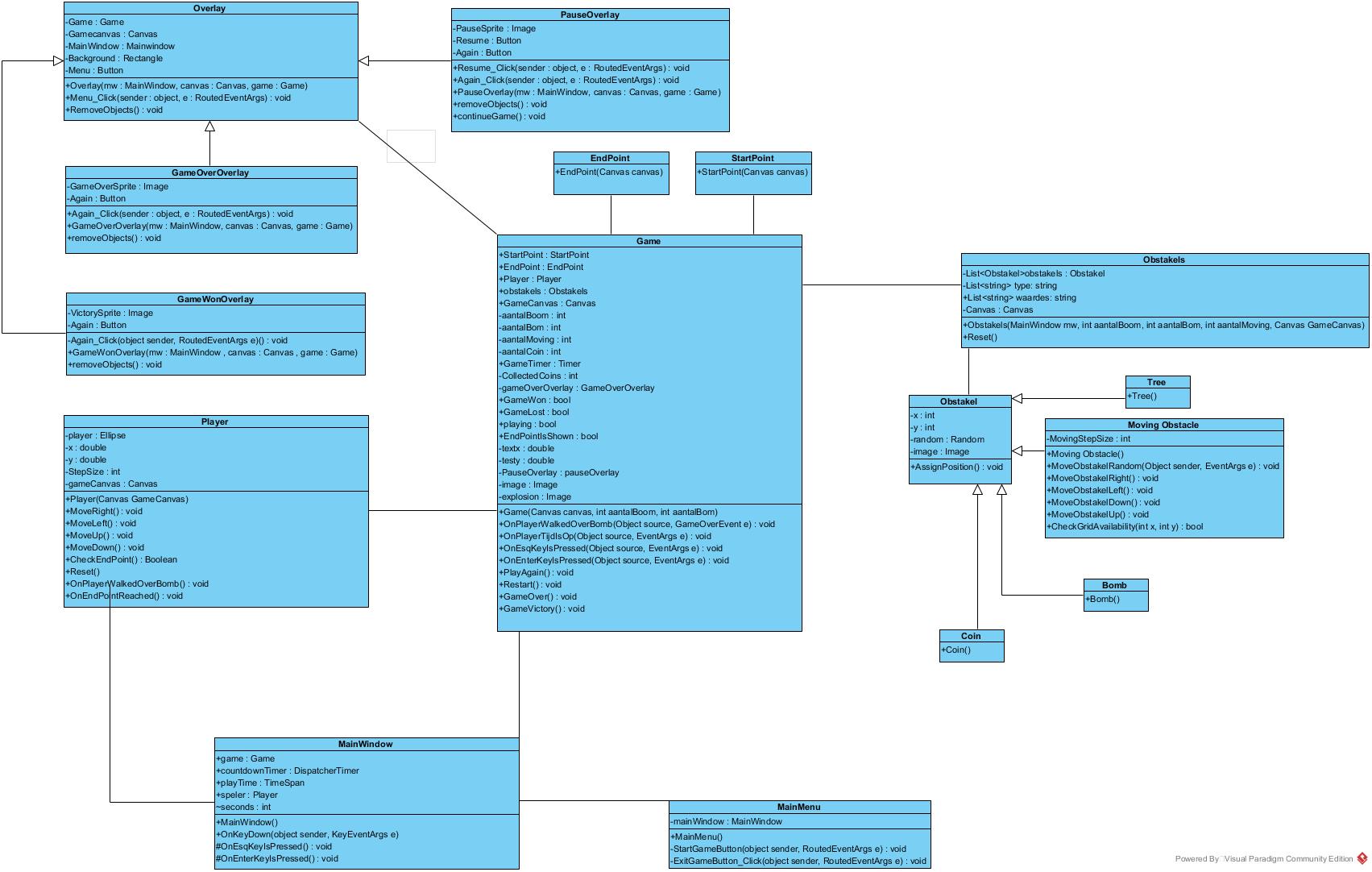
De game werkt grotendeels op events, eigenlijk gebeurt er pas wat als de gebruiker iets doet. De game doet bijna niks zelf. Pas als de gebruiker iets gaat doen zal het spel functionaliteit tonen. Als de speler op de pijltjes toetsen drukt zullen er verschillende functionaliteiten zijn, bijvoorbeeld dat de speler gaat bewegen. Wanneer dat het geval is wordt er ook direct gekeken of de speler tegen een boom, bom, spook of border aanloopt. Wanneer dat het geval is worden er weer nieuwe events aangeroepen, die zelf hun eigen functionaliteiten hebben.

Verder werken wij in deze game met timers. Deze timers worden op verschillende manieren en verschillende plekken toegepast. Wanneer de game wordt gestart begint er een timer te lopen om bij te houden wanneer het spel is afgelopen. Evenals loopt er een timer die ervoor zorgt dat een spook om de zoveel seconden van positie zal veranderen. Verder wordt er ook een timer gebruikt voor een mijn, om te kijken wanneer die moet ontploffen en wanneer de ontploffing gestopt is.

Ook hebben wij nog unit tests uitgevoerd. Voor veel verschillende onderdelen was het bijna niet mogelijk om unit tests te maken, mede omdat de functionaliteit daar niet geschikt voor was. Een andere reden is dat veel klassen samen in verband staan met elkaar, waardoor het geen unit tests meer zullen worden.

# Klassendiagram

In dit klassendiagram worden alle klassen overzichtelijk weergegeven en kun je makkelijker de relaties tussen klassen bekijken samen met de multipliciteit daartussen. Verder maakt het klassendiagram het overzichtelijk om te kijken welke methodes klassen bevatten samen met de attributen.



# Gebruik van events

In ons project hebben wij gebruik gemaakt van verschillende events om zo functies te kunnen activeren of gegevens kunnen verwisselen van verschillende klassen. Dit wordt in het klassendiagram beschreven. Om het allemaal duidelijker te maken volgt een uitleg over hoe alles werkt.

Zoals eerder al werd benoemd wordt in MainWindow een event subscriber gebruikt die zal bekijken wanneer er op een knop wordt gedrukt. Wanneer deze subscriber wordt geactiveerd zal het gaan kijken op welke knop er is gedrukt. Zodra het één van de pijltjestoetsen is zal er een methode worden aangeroepen in speler die de verdere afhandeling zal doen. Zodra er op de esc toets wordt gedrukt om de game op pauze te zetten wordt er een event geraised waar de klasse game op gesubscribed is. Wanneer dat het geval is zal de game op pauze worden gezet.

Wanneer de speler over een bom heen loopt wordt er ook een event aangemaakt in de klasse Player, omdat daar de functionaliteit staat. De klasse game subscribed op dit event en weet dus wanneer er op de bom wordt gelopen. In tegenstelling tot alle andere events bevat dit event informatie die wordt meegegeven aan de eventhandler. Deze informatie bevat de positie van de speler en de plek van de bom, dit is dan ook nodig om uiteindelijk de functionaliteit van het ontploffen van de bom te laten werken in de klasse Game.

Als er op een bom wordt gestaan, wordt de bom getoond en wordt er een timer geactiveerd. Deze timer raised een event na 1 seconde, na de 1 seconde wordt er gekeken of de speler nog in de buurt van de bom is (op dezelfde plek of 1 plek verwijderd van bom). Wanneer dat het geval is is het game over en wordt de gameover methode aangeroepen. Tijdens het event dat de ontploffing plaatsvindt wordt er nog een event aangeroepen die weer na 1 seconde een event raised. Na die seconde wordt de ontploffing weggehaald en is er geen mijn meer te vinden.

Wanneer er een bom wordt geactiveerd (er wordt dus op een bom gestaan) is het niet mogelijk dat er nog een bom wordt aangeroepen. Dit heeft er alles mee te maken dat er dan 2 events tegelijkertijd aan worden gemaakt wat ervoor zal zorgen dat de game zal crashen.

***<Hier moeten jullie de events beschrijving, dus hoe het werkt, wie er op gesubscribed is en wat het doet>***

# Sequentiediagram

Om voor bepaalde functionaliteit het overzicht te bewaren is het ook belangrijk om te kijken hoe bepaald klassen en methodes met elkaar samenwerken. Hiervoor hebben wij sequentiediagrammen gemaakt om dat overzichtelijk te houden.

