Ontwikkel document

Project: APEP

Groep: Team E

Auteurs: Pawel Zawada 18070035
Mikolaj Wichrowski 18136753
Tiko Vergroesen 14045230

Versie	Changelog	Datum
0.1	Eerste opzet met: ● requirements ● eerste versie van het klassendiagram ● toelichting op het klassendiagram	02/06/2019
0.2	Verbeterde opzet met: ● Genummerde requirements ● Verbeterde requirement beschrijvingen ● Nieuwe versie 0.2 UML ● Feedback	09/06/2019
0.3	Versie met: ● Referentie naar tests ● Nieuwe versie 0.3 UML ● Requirements voldaan toegevoegd	05/06/2019
0.4		

Inleiding

Dit document bevat de requirements analyse van de projectgroep Team E. De requirements van dit document zijn deels uit de casus gehaald en deels gemaakt als afspraak tussen de individuele leden van de projectgroep als logischerwijze aanvullingen op punten die goed aansluiten op de expliciete eisen die in de casus staan. De requirements zijn ingedeeld als functionele requirements en niet functionele requirements, waarbinnen ze verder zijn ingedeeld als “Must have” en “Nice to have”.

Requirements

De requirements van deze opdracht zijn:

Functionele eisen

Must have

1. De speler kan over de x en y as van het scherm bewegen. Deze beweging verloopt per vlak naar aanliggende velden.
2. De speler kan sleutels oppakken die op een veld vlak kunnen liggen.
3. De speler kun deuren openmaken met een eerder opgepakte sleutel.
4. Muren en deuren weerhouden de speler om er doorheen te lopen.
5. Het spel bevat ten minste één level.
6. Vooraf gemaakte kaarten kunnen meerdere keren gespeeld worden.
7. De speler kan niet van het speelveld af lopen.
8. Melding tonen wanneer level voltooid is.
9. Het level moet opnieuw op te starten zijn.
10. De speler kan maar 1 sleutel per keer oppakken.
11. Er kunnen meerdere sleutels van hetzelfde type op het veld liggen.

Nice to have

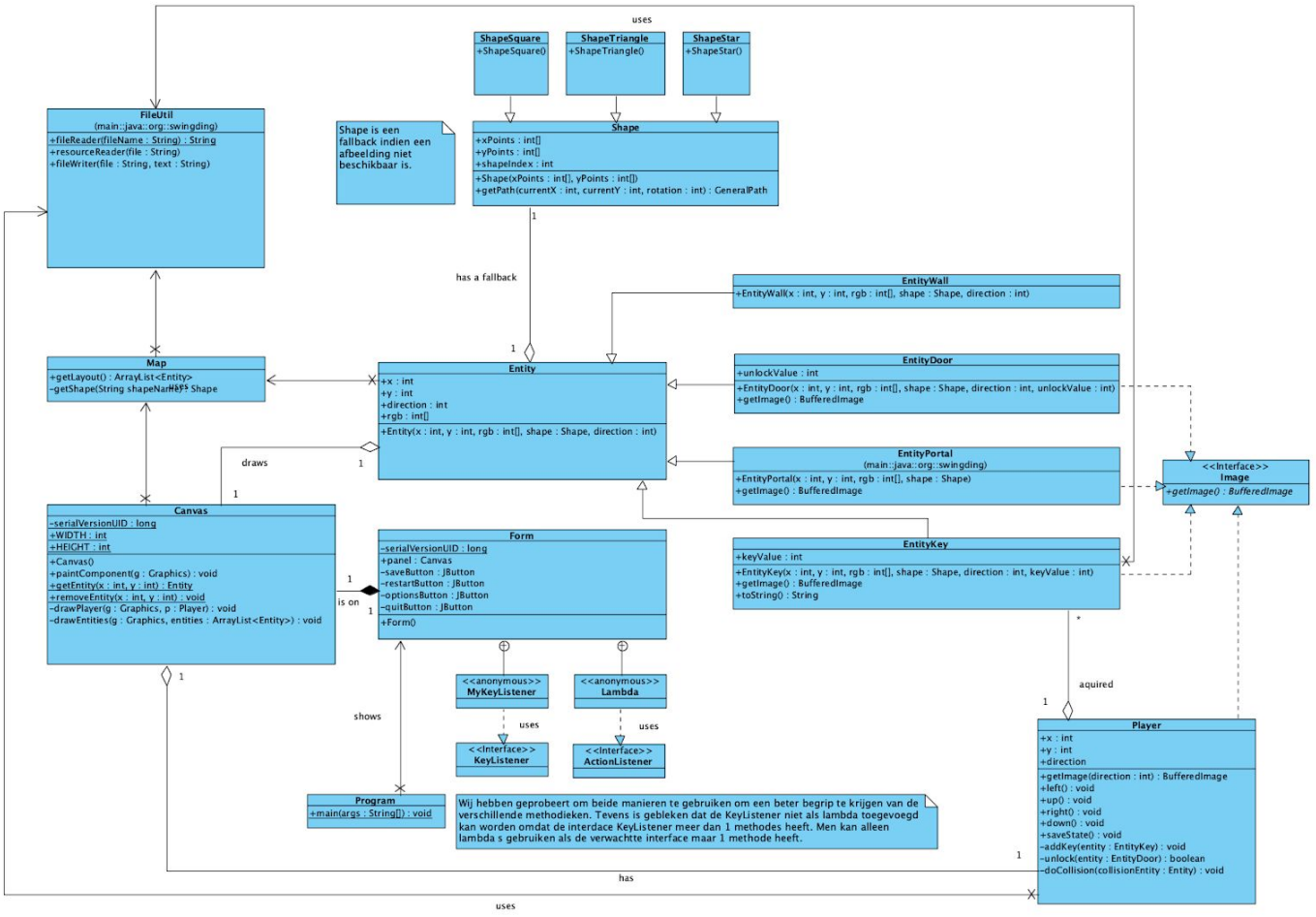
12. Levels worden automatisch gegenereerd op een willekeurige manier.
13. De speler toont beweeg animaties.
14. De speler keert naar de kant toe waar die naartoe loopt.
15. Voortgang tijdens het spelen van een level kan opgeslagen worden om later hervat te worden.
16. Speler kan kiezen uit meerdere levels die die op eigen wil kan spelen.
17. Het hele spel moet opnieuw te starten zijn.

Eisen voldaan

Must have: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11

Nice to have: 14, 15, 17

Klassendiagrammen



Toelichting

De `Canvas` klasse dient als de basis voor de interactie tussen de speler en het level waarin de speler bepaalde uitdagingen moet oplossen. In deze zullen alle `Entity` klassen plaatsvinden die dienen als de variabelen van elke uitdaging in de levels. Hij beheert het collisie systeem waarmee wordt verzekerd dat de speler niet door bepaalde obstakels heen kan, bijvoorbeeld om niet uit het level te lopen of om een puzzel op te lossen.

Voorwerpen zoals muren (`EntityWall`), deuren (`EntityDoor`) en sleutels (`EntityKey`) behoren allemaal toe aan hun parent klasse `Entity`, die als basis dient voor alle mogelijke varianten van een `Entity` die bepaalde karakteristieken deelt met zijn kinderen. Een `Entity` zal altijd een positie, vorm en directie hebben. Deuren kunnen worden geopend met een bijbehorende sleutel, door middel van de kombinatie van `EntityDoor.unlockValue` & `EntityKey.keyValue` attribute. De `Player` klasse deelt ook dezelfde basis attributen van positie en directie, maar beschikt ook over de mogelijkheid om met een interactie aan te gaan met bepaalde entiteiten, zoals deuren en sleutels. De `Image` klasse dient voor het opslaan van afbeeldingen, die dienen als visuele representaties van bepaalde entiteiten in het spel, zoals de speler, sleutels en deuren. `ShapeStar` toegevoegd om een ster te tonen als vorm. De nieuwe `Map` klasse dient als level voor het inladen en opslag van de verschillende levels, zodanig dat zij in zich, muren en deuren kunnen inhouden. De `Player` klasse heeft nu ook dependencies met `EntityKey` en `Canvas`. De `Player` klasse kan middels de `FileUtil` zijn afbeelding ophalen. De `EntityKey` klasse kan middels de `FileUtil` zijn afbeelding ophalen. De `EntityDoor` klasse kan middels de `FileUtil` zijn afbeelding ophalen. Wij maken gebruik van lambda's en anonymous klassen voor de event listeners. Zo hebben wij voor de knoppen/action listeners, de lambda's gebruikt. Voor de toetsenbord interactie, hebben wij de `MyKeyEvent` classe gebruikt. Deze wordt als anonymous classe aangemaakt door de java runtime en implementeert de `KeyEvent` interface.

Tests

Eis	Test	Toelichting
1	PlayerTest.class > class PlayerTest() > void upTest() PlayerTest.class > class PlayerTest() > void downTest() PlayerTest.class > class PlayerTest() > void leftTest() PlayerTest.class > class PlayerTest() > void rightTest() **Resultaat: geslaagd**	De tests kijken of de player na het bewegen een nieuwe locatie heeft. Gemaakt door Tiko
2	PlayerTest.class > class PlayerTest() > void tryToWalkIntoKeyTest() **Resultaat: geslaagd**	De test kijkt of de player een sleutel heeft na het aanraken van een sleutel. Gemaakt door Mikolaj
3	PlayerTest.class > class PlayerTest() > void tryToWalkIntoDoorWithKeyTest() **Resultaat: geslaagd**	De test kijkt of de player een deur kan openen met sleutel. Gemaakt door Mikolaj
4	PlayerTest.class > class PlayerTest() > void tryToWalkIntoWall() PlayerTest.class > class PlayerTest > void tryToWalkIntoDoorWithoutKeyTest() **Resultaat: geslaagd**	De tests kijken of de player op dezelfde locatie blijft staan als hij door de muur heen probeert te lopen of door een deur zonder de sleutel. Gemaakt door Mikolaj
7	FormTest.class > class FormTest() > void levelBoundaryTest() **Resultaat: geslaagd**	De test kijkt of de speler van het veld af kan lopen. Gemaakt door Mikolaj
9	FormTest.class > class FormTest() > void reloadTest() FormTest.class > class FormTest() > void restartTest() **Resultaat: geslaagd**	De tests kijken of het spel herstart is en of het level herstart is. Gemaakt door Mikolaj
15	PlayerTest.class > class PlayerTest > void loadStateTest() PlayerTest.class > class PlayerTest > void saveStateTest() **Resultaat: geslaagd**	De test kijkt of de gegevens van de laatste locatie en kaart van de speler opgeslagen worden. Gemaakt door Pawel

Wij hebben geprobeerd de meest relevante functionele eisen te testen door middel van unit tests. Deze hebben wij opgesteld en de auteur onder de test benoemd. Om er zeker van te zijn dat de relevante functionele eisen zijn getest en dat deze correct zijn hebben wij deze moedwillig laten falen en slagen. Zo konden wij oordelen over de effectiviteit en nuttigheid van de test.

Feedback

Docent	Opmerking	Datum
Steven	● Nummer de requirements. ○ Nummering van requirements toegevoegd. ● Klassendiagram volgens de UML notatie is vereist. ○ Klassendiagram aangepast. ● Technische eisen is niet verplicht. ○ Technische eisen verwijderd. ● “Geen god klasse” is niet een requirement. ○ Verwijderd uit requirements.	02/06/2019
Steven	● KeyListeners moeten in UML. ○ Deze als anonieme klasse benoemd. ● Toelichting moet meer zijn. ○ Toelichting aangepast.	11/06/2019

Todos laatste oplevering

- Save state verbeteren. Deuren worden bij het herstarten van het spel getekend. Wij moeten bijhouden welke deuren al zijn geopend en deze moeten niet meer getekend worden.
- Laatste test scripts moeten worden toegevoegd.
- Alle functionele eisen moeten worden nagelopen om er zeker van te zijn dat wij voldoen aan de functionele eisen.
- Laatste feedback verwerken van de docent.