**Testrapport**



Naam Project: Rust invaders // Gildaga

Naam product: testrapport

Datum: 22/12/2021

Schrijver: Koen Sampers

Klas: EI43AAPONE1RB

OV: 127437

Versienummer: 0.0.1

InhoudsOpgave

[Testplan 3](#_Toc91087075)

[Planning van de realisatie 3](#_Toc91087076)

[Testscenario’s 3](#_Toc91087077)

[Testlog 5](#_Toc91087078)

# Testplan

|  |  |
| --- | --- |
| Functionaliteit | Actie |
| Gebruiker gegevens opslaan | Slaat de gegevens op van een gebruiker |
| Voortbrengen // laten verschijnen van wezen | Brengt een wezen naar voor in het spel, laat een wezen verschijnen |
| Voortbrengen // laten verschijnen van speler object | Brengt de speler naar voor in het spel, laat de speler verschijnen |
| Speler moties // beweging | De mogelijkheid om de speler te bewegen op de X as in het speelveld. Met de volgende sneltoetsen; -A -D -🡨 -🡪 |

# Planning van de realisatie

|  |  |  |
| --- | --- | --- |
| Scene | Formulier | Functie |
| Main scene | Nee | Dit is de scene waar alles gebeurt op het speelveld. De speler heeft 3 levens en verdient punten door wezens neer te knallen. Wezens kunnen ook schieten en als je afgeknalt wordt krijg je een leven er af en ontstaat er een tussenpoos. Na deze tussenpoos laat de speler zich weer tevoorschijn toveren. Als de speler geen levens meer heeft wordt die verwezen naar de gameover scene. |
| Gameover scene | Ja | Hier kan de speler zijn gebruikergegevens opslaan. |

# Testscenario’s

|  |  |  |  |
| --- | --- | --- | --- |
| Functionaliteit: | Gebruiker gegevens opslaan |  |  |
| Actie |  |  |  |
| Scenario |  |  |  |
| Invoervelden: | Gebruikersnaam | Highscore |  |
|  | mrcoolguy21 | 100 |  |
| Verwacht resultaat: | Slaat de gegevens op van een gebruiker |  |  |
|  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Functionaliteit: | Voortbrengen // laten verschijnen van wezen |  |  |
| Actie |  |  |  |
| Scenario |  |  |  |
| Invoervelden: |  |  |  |
|  |  |  |  |
| Verwacht resultaat: | Brengt een wezen naar voor in het spel, laat een wezen verschijnen |  |  |
|  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Functionaliteit: | Voortbrengen // laten verschijnen van speler object |  |  |
| Actie |  |  |  |
| Scenario |  |  |  |
| Invoervelden: |  |  |  |
|  |  |  |  |
| Verwacht resultaat: | Brengt de speler naar voor in het spel, laat de speler verschijnen |  |  |
|  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Functionaliteit: | Speler moties // beweging |  |  |
| Actie |  |  |  |
| Scenario |  |  |  |
| Invoervelden: | gebruikerInput |  |  |
|  | A, 🡨, D, 🡪 |  |  |
| Verwacht resultaat: | Speler kan van links naar rechts bewegen in het speelveld op de X as. |  |  |
|  |  |  |  |

# Testlog

|  |  |
| --- | --- |
| Scene: | Main scene |
| Datum test: | 18/12/2021 |
| Tester: | Koen Sampers |
| Probleem: | Ervoor zorgen dat een gameover kan plaats vinden. |
| Prioriteit: | 3 |
| Verbeteractie: | //Er kunnen geen enemies en player objecten meer verschijnen in het //speelveld, dus in player.rs en enemy.rs staat het volgende:  if game\_state.0 != "gameover".to\_string(){  //spawn object  //. . .  }  //De *game\_state.****0***attribuut is een speciale attribuut die bestaat in een //unique type in rust genaamd *tuple struct.*  //GameState ziet er zo uit:  struct GameState(String);  // De **0** attribuut is de string die genest zit in de GameState. Dit is voor structs // waar de naam een field van zelfsprekend is. |
| Datum actie: | 19/12/2021 |
| Afhandelaar: | Koen Sampers |
|  |  |

|  |  |
| --- | --- |
| Scene: | Alle scenes |
| Datum test: | 20/12/2021 |
| Tester: | Koen Sampers |
| Probleem: | Nu worden alle processen groten deels in de main scene gedaan, het zou fijn zijn als alle verschillende functionaliteiten m.b.t. de systemen, plugins en entiteiten binnen de applicatie in aparte custom plugins gestopt kunnen worden. Hierdoor wordt de code een stuk leesbaarder |
| Prioriteit: | 1 |
| Verbeteractie: | //Als voorbeeld bovenaan in main.rs  use crate::player::PlayerPlugin;  //crate geeft het pad aan in je eigen project, player staat voor de bestand //player.rs en in dat bestand kun je de public struct PlayerPlugin vinden  //Deze custom plugin kun je gelijk toevoegen aan main.rs:    .add\_plugin(PlayerPlugin);  //Voorheen was het dus dat alle code van deze PlayerPlugin gewoon zo los //stond in main.rs, en dat geld niet alleen voor de PlayerPlugin;  .add\_plugin(PlayerPlugin)  .add\_plugin(EnemyPlugin)  .add\_plugin(WindowPlugin)  .add\_plugin(UiPlugin)  .add\_plugin(StatePlugin) |
| Datum actie: | 21/12/2021 |
| Afhandelaar: | Koen Sampers |
|  |  |

|  |  |
| --- | --- |
| Scene: | Gameover scene |
| Datum test: | 16/12/2021 |
| Tester: | Koen Sampers |
| Probleem: | Er is nog geen weergave voor de errormessage van de database. Dit is nodig om te weten of er iets fout gaat met de database connectie. |
| Prioriteit: | 2 |
| Verbeteractie: | //Voor context geef ik dit mee om het probleem beter uit te leggen  //Deze future object is nodig om de create query en database connectie async en dus op //een thread apart van de applicatie uit te voeren.  let future = save\_to\_db(&player\_state.username, player\_state.score);  //hier roept die de save\_to\_db() functie op een aparte thread aan met de block\_on() //functie  let result = block\_on(future);  // --NIEUW—Check of de result object een SqlError heeft meegegeven.  match result{  Ok(\_) => player\_state.username = "score saved!".to\_string(),  Err(\_) => player\_state.username = "something went wrong".to\_string()  }  //Deze functie maakt een database connectie aan en voert ook gelijk de create query uit.  async fn save\_to\_db(username: &str, score: u32) -> Result<(), sqlx::Error>{  //maak nieuwe connective aan  let pool = MySqlPoolOptions::new().max\_connections(5).connect("mysql://localhost/gildaga").await?;  // voer query uit  sqlx::query("INSERT INTO score (Username, Score) VALUES ( ?, ? )").bind(username).bind(score).execute(&pool).await?;  //Als alles goed is gegaan return Ok()  Ok(())  } |
| Datum actie: | 17/12/2021 |
| Afhandelaar: | Koen Sampers |
|  |  |

|  |  |
| --- | --- |
| Scene: | Main scene |
| Datum test: | 19/12/2021 |
| Tester: | Koen Sampers |
| Probleem: | Er voor zorgen dat de inspector\_egui library werkt |
| Prioriteit: | 0 |
| Verbeteractie: | //Om code te debuggen in bevy met rust bestaat er een inspector\_egui library.  //Dit kun je gebruiken om objecten in te zien, met alle attributen die aan een //entiteit zitten.  //Met deze line geef je aan dat je een object wilt inspecteren:  #[derive(Inspectable, Default)]  struct Enemy;  //Nu kun je de inspectorplugin aanroepen om alle enemy entiteiten in de //inspector\_egui te registreren:  .add\_plugin(InspectorPlugin::<InspectorQuery<(Entity), With<Enemy>>>::new());  //De bovenste line duid: *Pak alle entiteiten die een Enemy component bevatten* //*en stop deze in de InspectorPlugin.*  //Resultaat: |
| Datum actie: | 20/12/2021 |
| Afhandelaar: | Koen Sampers |
|  |  |

Verschillende categorien binnen de prioriteiten die gesteld zijn:

Prioriteit 3 = heel hoog

Prioriteit 2 =hoog

Prioriteit 1 = normaal

Prioriteit 0 = laag