Tussen backticks als deze staan cues voor het invulling geven aan de topics. Verwijder ze uiteindelijk. Dit ontwikkeldocument is bedoeld als overdrachtsdocument en naslagwerk, voor de opdrachtgever, voor eventuele toekomstige teams die verder werken aan het product, maar ook tijdens de ontwikkeling voor het huidige team. Probeer de huidige staat van de ontwikkeling altijd zo goed mogelijk gesynct te houden met het ontwikkeldocument, zodat die ook helder is voor alle teamleden. Belangrijke Tip: Een valkuil om op verdacht te zijn, is dat het ontwikkeldocument een losse verzameling van ingevulde hoofdstukjes wordt. Dat willen we dus niet. Er dient overal voldoende en heldere tekst toegevoegd te zijn die logica en samenhang (zoals tussen de opeenvolgende hoofdstukken, of voor wat betreft gemaakte keuzes) uitlegt. Tot en met het ontwerp moet het zonder extra mondelinge toelichting duidelijk, samenhangend en makkelijk leesbaar zijn voor een gemiddelde persoon met slechts lichte it-kennis. Een deel van de ontwikkeldocumentatie is afgesplitst van dit document: documentatie t.a.v. het ontwerp en realisatie van het web-subysteem (flask, mongodb, html, css, javascript) is afgesplitst in een apart, ietwat ander type ontwikkeldocument. Ander belangrijk ding: gebruik alleen links naar publieke websites of relatieve links (dus binnen de team-repo, NIET naar je persoonlijke repo), zodat uiteindelijk uit de team-repo een zip gemaakt kan worden voor de opdrachtgever, waarvan de links (nog) werken.

Zie plantuml tutorial in S3 voor de tutorial in S3 ove plantuml.

Ontwikkeldocument Project `projectnaam`

Versie `1.1.1` Naam `Ruben`

Inhoudsopgave

Inleiding

De opdracht komt van de HU in de HBO-ICT opleiding en wordt aan studenten gegeven om meer skills te ontwikkelen bij het gebruiken van c++ in vscode. De opdracht gaat over het maken van een game en het maken van diagrammen en modellen die niet van de code afwijken. Het document wordt gebruikt om overzicht en samenhang te creëren voor het maken van het game project. In dit document wordt veel informatie opgeschreven over het project om zelf bij de kunnen houden hoeveel nog gedaan moet worden en om feedback op het proces en tekst te kunnen vragen.

`Van wie komt de opdracht? Waar gaat de opdracht in hoofdlijnen over? Leg verder uit dat dit document bedoeld is om op heldere wijze overzicht en samenhang te geven voor het team tijdens het werken aan het project, en na afloop als overdrachts-document voor eventuele follow-ups.`

Leeswijzer

In het uitgangspunten hoofdstuk worden het systeem context, de key drivers en de requirements besproken over mijn game project. Het systeem context geeft een breed idee waar het systeem van de game uit bestaat en welke actoren ermee te maken hebben. De key drivers geven duidelijk aan welke hoofdredenen er zijn om de game te maken. De requirements geven aan waar de game aan moet voldoen en op welke manier wordt in de non-functionele requirements aangegeven. `leg uit wat er in de hoogste-niveau-hoofdstukken wordt behandeld en hoe deze onderwerpen met elkaar in verband staan`

Uitgangspunten

Hier wordt uitgelegd waarom de speficieke requirements van belang zijn en vanaf welke basis de game gemaakt gaat worden. Hier een verslag van genoteerde informatie van de opdrachtgever. dus schrijf hier op welke input ik tijdens gesprekken van docenten heb gekregen

`Leg uit dat dit hoofdstuk de uitgangspunten voor de requirements inventariseert. Verwijs naar een appendix met genoteerde input (verslag van speech, gespreksverslagen) van de opdrachtgever - (de echte, tijdens de kickoff-meeting of diens vervanger (Marius,Bart) erna)`

Systeem Context

Ik heb eerst gekeken welke actoren te maken hebben met het systeem: ik kwam op een opdrachtgever, developer, gebruiker en concurrent. Het bestaat uit deze actoren, de game, de knoppen, de esp-32 en de laptop. Op de pijlen staan wat de relaties tussen de systemen en actoren zijn. De game runt op de esp-32 en deze wordt gerunt met de laptop. De knoppen worden geregistreerd en dan stuurt het input naar de game waardoor het karakter hierop reageert.

Modelleer en beschrijf de Systeem Context – gebruik een systeem context diagram. Onderdelen van binnen je System of Interest horen er niet in thuis. Een decompositie van je systeem ook niet. Zijn er geen belangrijke actoren vergeten? Tip: maak eerst met je team een mindmap (zie appendix) alt text

Identificatie en prioritering van Key Drivers

Bij dit hoofdstuk heb ik feedback aan Marius gevraagd over mijn key drivers tabel. Hij heeft gezegd dat het niet van belang is om een key drivers tabel te hebben over het spel. Nu houd ik het bij een simpel key drivers tabel dat niet heel diep gaat. Hij zei dat ik meer aandacht moet geven aan het maken van use cases

Bij het ontwikkelen van een game is het misschien handig om te weten welke stakeholders erbij betrokken zijn en welke key drivers zij hebben. In dit project worden vier stakeholders kort onderzocht. Van elke stakeholder wordt kort beschreven wat zijn rol en belangrijkste motivatie is, om duidelijk te maken waarom zij met de game te maken hebben.

Hier de key drivers voor het game project in een tabel:

Stakeholder	Beschrijving	Key drivers
opdrachtgever	De opdrachtgever is de docent of een bedrijf dat opdrachten aan studenten geeft.	outreach, financieel rendement, resultaat en toekomstig talent ontwikkelen
developer	de developer is de persoon die de game individueel of in teamverband ontwikkelt.	inkomsten, inzichten gebruikersdata, passie voor creëren, community opbouwen
gebruiker	de gebruiker is de persoon die de game koopt, inlogt met een account en speelt.	plezier, sociale verbondenheid, uitdaging, positieve spelervaring
concurrent	de concurrent is de persoon of bedrijf dat een game gemaakt heeft dat zich in de dezelfde genre bevind waardoor gebruikers moeten afwegen om het spel van de concurrent of van de developer te kopen en te spelen.	marktpositie versterken, meer spelers aantrekken, winst, datagedreven optimalisatie

Voor semester 3 beperken we ons tot 3 stakeholders: opdrachtgever, klant en gebruiker. Geef middels een tabel een overzicht van de key drivers weer. Belangrijkere key drivers staan hoger in de tabel. Er is ook een kolom die aangeeft voor welke stakeholder het van

toepassing is, en een kolom voor omschrijving. Licht de ordening van de key drivers toe in de begeleidende tekst.

Use Cases

De opdracht gaat over dat je een karakter bestuurd waarmee je obstakels wilt ontwijken en munten wilt pakken. Met mijn groepsgenoten heb ik bepaald wat het minimum valuable product van mijn spel gaat zijn. Mijn karakter moet kunnen bewegen en punten pakken. Objecten zoals vierkanten worden willekeurig gespawnd vanaf de buitenkant van het scherm die naar binnen kunnen bewegen en daarna weer verdwijnen. Bij het karakter wordt actief gecheckt of het een coin aanraakt en of het karakter objecten of projectiles aanraakt die een leven van je weghaalt. In dit document wordt bijgehouden wat ik allemaal voor het game project gedaan heb, ook naast de code.

Feedback van het game pitch idee dat ik nu heb. Functioneel 1: Ik ga meerdere knoppen gebruiken om de richting van een karakter te bepalen en nog een knop waardoor je tijdelijk niet af kan gaan of dat je een dash doet. de sensoren of actuatoren zijn niet complex genoeg voor boven niveau. score: 1 Functioneel 2: Ik ga een spel maken waarbij je met je karakter meerdere bewegende obstakels wilt ontwijken door te bewegen en snel te reageren, ik ga proberen veel objecten te maken die snel bewegen en dus niet alleen een paar grote objecten die sloom bewegen want dan is dat niet complex genoeg. misschien net complex genoeg omdat er veel verschillende grote en kleine objecten gebruikt kunnen worden om het spel lastig te maken. score: 1/2 Functioneel 3: Ik ga eerst de scoreboard op het scherm werkend krijgen en als ik meer tijd heb dan misschien bluetooth ook. score: 1

Een of meerdere use case diagram(men) met bijbehorende use case beschrijvingen

Naam	UC01 - Speler beheren
Actor	Developer
Doel	Het doel van de Speler beheren is om botsing te detecteren, de score en levens bij te houden.
Samenvatting	Deze use case beschrijft hoe de speler in de game werkt bij botsing en welke variabelen daarbij horen.
Preconditie	De speler is nog niet op het scherm gezet
Scenario.	1. De gebruiker drukt op de startknop om te beginnen. 2. De speler wordt op het scherm gezet. 3. De knoppen sturen de input naar de speler waardoor de speler beweegt 4. Als de speler klasse 0 levens heeft wordt de speler van het scherm afgehaald 5. Als de knoppen een signaal sturen dat de game weer opnieuw moet runnen dan wordt de speler weer geactiveerd.
Invariant	1. Als de speler geraakt is heeft de speler tijdelijke onkwetsbaarheid en gaat er 1 leven van de levens af. 2. Als de speler een munt opgepakt heeft, dan wordt bij de score 1 opgeteld. 3. De knoppen sturen altijd de inputs naar de speler zodat de speler snel van richting kan veranderen.

Naam	`UC01 - Speler beheren`
Postconditie	De speelsessie is beëindigd doordat de speler geen levens meer heeft en wordt de speler van het scherm af gehaald.
Uitzonderingen	n.v.t
Naam	`UC02 - Enemies beheren`
Actor	Developer
Doel	Het doel van de Enemy beheerder is om de snelheid, positie en projectielen van enemies te regelen en ervoor zorgen dat enemies gespawned worden en gedespawned worden.
Samenvatting	Deze use case beschrijft hoe de enemy mechanics werken in de game
Preconditie	Er staan geen enemies op het scherm
Scenario.	1. Als er geen enemies op het scherm staan dan wordt een enemy gespawned, deze beweegt een kant op. 2 De enemy verlaat het scherm en wordt buiten het scherm geplaatst en gereset. 3. De enemy kan weer opnieuw gebruikt worden met andere waardes voor de positie of de projectielen, keer terug naar stap 1.
Invariant	1a. Als de speler met een enemy botste, wordt een bericht naar die enemy gestuurt en verdwijnt die enemy van het scherm.
Postconditie	De enemy wordt van het scherm afgehaald en gereset als de speelsessie is beëindigd, de enemy het scherm verlaat of het botste met de speler.
Uitzonderingen	n.v.t.
Naam	`UC03 - Spawn Enemy1`
Actor	Developer
Doel	Het doel van Enemy1 is om het moeilijker te maken voor de speler om de game te overleven. Dit doet Enemy1 door te bewegen en projectielen te schieten
Samenvatting	Deze use case beschrijft hoe Enemy1 mechanics werken in de game
Preconditie	Enemy1 wordt op het scherm gespawned
Scenario.	1. Enemy1 beweegt een kant op in een rechte lijn en schiet projectielen naar voren. 2 Enemy1 blijft bewegen over het scherm totdat hij het scherm verlaat of tegen de speler botst. 3 Enemy1 kan weer opnieuw gebruikt worden met andere waardes voor de positie, de snelheid of de projectielen, keer terug naar stap 1.
Invariant	1a. Als de speler met Enemy1 botste, dan verdwijnt die Enemy1 van het scherm. 2a. Enemy1 blijft checken of hij buiten het scherm is of tegen de speler botst.
Postconditie	Enemy1 botst met de speler of gaat van het scherm af en verdwijnt.
Uitzonderingen	n.v.t.

Naam	`UC04 - Spawn Projectiel`
Actor	Developer
Doel	Het doel van een projectiel is om de speler het projectiel te laten ontwijken door te bewegen.
Samenvatting	Deze use case beschrijft hoe een projectiel in het algemeen in het spel gaat werken
Preconditie	Het projectiel is klaar om te gebruiken en staat nog niet op het scherm afgebeeld
Scenario.	1. Een projectiel wordt vanaf de buitenkant van het scherm gespawned of vanaf een enemy geschoten. 2. het projectiel beweegt een kant op met een snelheid 3. Als het projectiel met de speler botst, wordt een bericht naar het specifieke projectiel gestuurd en wordt het projectiel van het scherm afgehaald. 4. De snelheid en positie van het projectiel worden gereset zodat het projectiel opnieuw gebruikt kan worden, keer terug naar stap 1.
Invariant	1a. Als er projectielen boven de maximale waarde op het scherm staan, dan worden er niet meer toegevoegd tot weer een projectiel van het scherm verdwijnt. 3a. Als de speler met een projectiel botst dan wordt een bericht gestuurt dan ze met elkaar gebotst hebben.
Postconditie	Het projectiel wordt van het scherm afgehaald en gereset als de speelsessie is beëindigd, als het projectiel het scherm verlaat of als het botste met de speler.
Uitzonderingen	n.v.t.
Naam	**``UC05 - Spawn Munt`**
Actor	Developer
Doel	De munt kan door de speler opgepakt worden, zo kan de gebruiker een score krijgen voor zijn run.
Samenvatting	De munt wordt afgebeeld op het scherm en kan opgepakt worden om 1 aan de score toe te voegen.
Preconditie	De game staat aan en wacht op input van de gebruiker.
Scenario.	1. Spawn een munt op het scherm, steeds op een positie anders dan de vorige. 2. Als de speler botst met de munt, wordt de munt van het scherm afgehaald. 3. Bij de score wordt 1 opgeteld, keer terug naar stap 1.
Invariant	2a. Er wordt altijd gecheckt of de speler de munt aanraakt.
Postconditie	De munt is opgepakt en wordt op een andere plek gespawned. Pas bij game over wordt de munt permanent van het scherm afgehaald.
Uitzonderingen	n.v.t.
Naam	`UC06 - Game Spelen`
Actor	Gebruiker

Naam	UC06 - Game Spelen
Doel	De gebruiker wil de game starten, spelen en een score behalen.
Samenvatting	Deze use case beschrijft hoe de gebruiker de game opstart, speelt tot alle levens op zijn en vervolgens het eindresultaat ziet.
Preconditie	De game staat aan en wacht op input van de gebruiker.
Scenario.	1. Wacht totdat de "voorwaarts" knop wordt ingedrukt. 2. Zet de waardes van score op 0, levens op 3. 3. Spawn de speler, projectielen en enemies. 4. De speler beweegt zich. 5. Obstakels en projectielen verschijnen op het scherm en bewegen. 6. Stuur een bericht als er botsing plaatsvind tussen de speler en een projectiel vanaf de speler beheren use case naar de use case van de enemy of projectiel. 7. Als het derde leven op is dan komt "Game over" en de score op het scherm te staan en stopt de game loop. 8. Keer terug naar stap 1.
Invariant	1a. Als de "voorwaarts" knop ingedrukt wordt bij het beginscherm, dan wordt de game opgestart. 4a. alle knoppen blijven signalen sturen naar de speler als ze ingedrukt worden. 6a. De speler klasse stuurt een signaal als het derde leven op is om "Game over", de score op het scherm te zetten en score en levens resetten voor een nieuwe run.
Postconditie	De speelsessie is beëindigd. De eindscore is zichtbaar en de gebruiker kan opnieuw beginnen door weer op de "voorwaarts" knop te drukken.
Uitzonderingen	Als de game handmatig gestopt wordt dan is de game loop niet meer oneindig.

Activity Diagrammen

Voor belangrijke, wat complexere usecases kan het extra verduidelijking opleveren om er SysML - activity diagrammen bij te maken.

Ontwerp

leg uit hoe het ontwerp volgt door het volgen van de stappen die in de sub-hoofdstukken worden behandeld en hoe deze onderwerpen met elkaar in verband staan

Functionele decompositie / (sub)systems and interfaces

Geef met een functionele decompositie van het systeem grafisch weer hoe de verschillende functies (uit de functionele requirements) van het systeem met elkaar samenhangen. Dat geeft goede handvaten voor de decompositie van software en hardware verder op de rit.

Objectmodellen

Ontwerp, uitgaande van use case beschrijvingen en activity diagrammen, de delen van het objectmodel die dat kunnen waarmaken. Beperk de interactie met het web-subsysteem tot

een enkel object (een web proxy object oid). Een gedetailleerde uitwerking van het web-subsysteem is in een apart ontwikkeldocument te vinden.

Lijst met Objecten

Voeg elk object uit de objectmodellen toe in de "lijst met objecten" Let op dat de beschrijvingen niet de relatie tussen de objecten duiden, maar louter wat objecten "los bekeken" doen. Dus niet: InstelControl stuurt een signaal naar .. Maar Instelcontrol is de "dirigent" van de usecase "Instellen" (meteen link toevoegen naar die usecase).

Object Naam	Stereotype	Beschrijving
PlayerControl	Control	"Dirigent" van de use case "Speler beheren"
EnemyControl	Control	"Dirigent" van de use case "Enemies beheren"
ProjectileControl	Control	"Dirigent" van de use case "Spawn Projectiel"
GameControl	Control	"Dirigent" van de use case "Game Spelen"
ButtonControl	Control	"Dirigent" van de knoppen hardware
Display	Boundary	stuurt de display hardware aan
Buttons	Boundary	stuurt de knoppen hardware aan
CoinEntity	Entity	bevat de informatie van de coin
EnemyEntity	Entity	bevat de informatie van de enemies
ProjectileEntity	Entity	bevat de informatie van de projectiles
GameSettings	Entity	bevat de informatie van de constante waardes
HeartSprite	Entity	bevat de informatie van de HeartSprite C array

Taakstructurering

leg uit wat het doel is van taakstructurering en hoe de deelstappen (sub-hoofdstukken) samen dat doel reliseren

Taaksoort en deadline

Maak een tabel die per object taaksoort, deadline, periode en prioriteit weergeeft. Belangrijk: Deadline is zo lang mogelijk waarbij het nog net geen irritatie oplevert. Deadline <= Periode, Prioriteit is omgekeerd evenredig met deadline

In onderstaand tabel heb ik gekeken naar hoe de taken bij elk object er uit zouden zien.

Taakstructureringstabel NS trein opdracht:

Objecten	Taaksoort	Periode	Deadline	Prioriteit
GameControl	Intern, Demand Driven		100ms	3

Objecten	Taaksoort	Periode	Deadline	Prioriteit
PlayerControl	Intern, Demand Driven		100ms	3
EnemyControl	Intern, Demand Driven		100ms	3
ProjectileControl	Intern, Demand Driven		50ms	2
ButtonControl	Intern, Demand Driven		50ms	2
Display	IO, Demand Driven		50ms	2
ForwardButton	IO, Periodical	30ms	50ms	1
BackwardButton	IO, Periodical	30ms	50ms	1
LeftButton	IO, Periodical	30ms	50ms	1
RightButton	IO, Periodical	30ms	50ms	1
AbilityButton	IO, Periodical	30ms	50ms	1
CoinEntity	Geen taak	-	-	-
EnemyEntity	Geen taak	-	-	-
ProjectileEntity	Geen taak	-	-	-
GameSettings	Geen taak	-	-	-
HeartSprite	Geen taak	-	-	-

Taken samenvoegen

`Maak een tabel waarin je laat zien welke objecten een eigen taak hebben en van welke de taken worden samenvoegd in een enkele "Taak". Noem in het laatste geval het object (bijvoorbeeld een handler) dat eigenaar wordt van die Taak als eerste.`

In onderstaand tabel heb ik nagedacht hoe ik de taken zou verdelen over de objecten en hoe ik ze samen kon voegen. Dit heb ik bij de lichtjes en de IO Periodical taaksoorten gedaan, omdat deze samengevoegd kunnen worden in twee Handler objecten

Taken samenvoegen tabel NS trein opdracht:

Taak	Objecten	Taaksoort	Periode	Deadline	Prioriteit
GameTaak	GameControl	Intern, Demand Driven		100ms	3
PlayerTaak	PlayerControl	Intern, Demand Driven		100ms	3

Taak	Objecten	Taaksoort	Periode	Deadline	Prioriteit
EnemyTaak	EnemyControl	Intern, Demand Driven		100ms	3
ProjectileTaak	ProjectileControl	Intern, Demand Driven		50ms	2
ButtonControlTaak	ButtonControl	Intern, Demand Driven		50ms	2
DisplayTaak	Display	IO, Demand Driven		50ms	2
ButtonTaak	ButtonHandler, ForwardButton, BackwardButton, LeftButton, RightButton, AbilityButton	IO, Periodical	30ms	50ms	1

Klassediagrammen

Ontwerp, uitgaande van de objectmodellen de bijbehorende klassediagrammen. Vergroot eventueel in latere verbeteringsronden de herbruikbaarheid en het gebruiksgemak van de klassen door het toepassen van geschikte Design Patterns of templating. Voeg de klassen ook toe in de Requirements Traceability diagrammen zodat duidelijk is welke requirements de klasse adresseert

STD's


GameControl


PlayerControl

Ontwerp voor elke Taak de STD van de bijbehorende klasse(n), indien van toepassing vanuit activity diagram of usecase beschrijving, protocol of anderszins. Belangrijk: alle toestanden moeten gerepresenteerd worden in het diagram. Code zonder toestanden en zonder directe invloed op de flow tussen de toestanden kunnen gerepresenteerd worden door calls naar helper-functies. Vergeet niet bovenaan een geschikte STD-interface toe te voegen

Realisatie

leg uit waarin de realisatie zich onderscheidt van het ontwerp, en noem kort de rollen van de subhoofdstukken

Fysieke View

Ontwerp de fysieke decompositie (een SysML Bdd). (de functionele decompositie biedt daarvoor meestal goede aanknopingspunten). Verduidelijk fysieke compositie-relaties middels "Constraints" (NB: dat zijn in dit geval gekwantificeerde ontwerpkeuzes die zowel uit de eerdere constraints als de niet-functionele requirements bestaan.

Code

Ontwerp vanuit de STD's de bijbehorende code. Geef waar nodig additionele toelichting. Voeg per STD een link toe naar de betreffende code in de team repo. Voeg ook links naar de overige code toe

Unit Tests

Voeg in dit hoofdstuk sub-hoofdstukken toe met Unit Tests, elk bestaande uit een Testplan, een link naar de testcode, een samenvatting van de bevindingen van de meest recente uitvoer van die test en een link naar een bestand met die meest recente test-output.

Integratie Tests

Voeg in dit hoofdstuk sub-hoofdstukken toe met Integratie Tests, elk bestaande uit een Testplan, een link naar de testcode, een samenvatting van de bevindingen van de meest recente uitvoer van die test en een link naar een bestand met die meest recente test-output. Belangrijke integratie-tests zijn uiteraard ook de tests van het complete product. Hoe goed doet het product wat het moet doen?

Verder zou je in dit hoofdstuk Realisatie ook subhoofdstukken kunnen opnemen met : beslissingstabellen, waarin gemaakte realisatiekeuzes worden verantwoord, Failure Mode Effect Analyses, en andere realisatie gerelateerde overwegingen.

Eindresultaat

Nog eens op een rijtje de behaalde functionaliteiten en de performance.

Conclusie en Advies

Reflecteer op in welke mate de aanvankelijk opgestelde requirements daadwerkelijk zijn behaald. Geef goed onderbouwde aanbevelingen t.a.v. mogelijke toekomstige doorontwikkeling.

Appendices

Appendix 1: Mindmaps

Bijvoorbeeld voorafgaand aan het maken van het systeem context diagram is een mooi moment om met zijn allen een mindmap te maken die een samenhang duidt van alles wat je als team maar kunt bedenken in relatie tot het systeem.

Appendix 2: Gespreksverslagen

Notities bij Kickoff-Meeting

... etc

Appendix 3: Upgradeonderzoeksverslagen

zet hier een lijst met links naar de upgradeonderzoeksverslagen. Licht het toe met wat tekst met de belangrijkste samenvatttingen ervan.

Appendix 4: Referenties

lijst 3rd party materiaal waar naar verwezen is. Boeken, site-links.

Verder mogelijk nog appendices over: de geschiedenis van de ontwikkeling van het product in grote lijnen (inclusief inmiddels afgeschoten tussenproducten), over het opzetten van de ontwikkelomgevingen, over hoe te debuggen. Houdt het bij samenvattende dingen. Lange teksten (zoals testuitvoer) niet in de appendices maar via links naar bestanden opnemen.