Taaru no Kizoku

OGO 1.1: Groep 1a

door Leroy Bakker 0617167
Roy Berkeveld 0608170
Giso Dal 0615787
Etienne van Delden 0618959
Edin Dudojević 0608206

19 Januari, 2007

Inhoudsopgave

Hoofdstuk 1

Samenvatting

Dit document bevat, naast de standaardonderdelen van het verslag, een uitwerking van een pacman-variant Taaru no Kizoku. Deze uitwerking bestaat uit de user requirements, een inventarisatie, een specificatie en een korte synthese. Het maken van dit project heeft inzicht verschaft in de leer van het formeel specificeren en het interpreteren van formele specificaties. Ook heeft het ons in aanraking gebracht met het inventariseren.

Hoofdstuk 2

Inleiding

Het doel van OGO is leren samenwerken aan een project dat qua omvang en benodigde tijd veel groter is dan bijvoorbeeld huiswerk opgaven. De kennis die bij de reguliere vakken is vergaard kan bij OGO in de "praktijk" worden toegepast.

In dit project hebben wij de pacman-variant "Taaru no Kizoku" uitgewerkt. Wij hebben op pacman een Japans thema gebombardeerd, waarbij Pacman een sumoworstelaar is, die sushi moet eten, wasabi power-ups heeft en samurai als tegenstanders heeft.

We gaan eerst de user requirements maken, met daarin hoe het spel er uitziet. Vervolgens een inventarisatie, die later gespecificeerd wordt.

Hoofdstuk 3

User Requirements

"Taaru no Kizoku" is een geschikt spel voor jong en oud! In deze, japans georiënteerde, single player pacmanvariant zul je de avonturen beleven van sumo worstelaar "Taaru no Kizoku". Je zult jezelf moeten manoeuvreren door de levels, door alle "sushi's" te eten die zich in dat level bevinden. Zorg dat je hierbij niet gepakt wordt door de gevaarlijke samurai, want die zullen ervoor zorgen dat een van je levens verliest ,die je in het begin van het spel krijgt. Dan begint het level dus gewoon opnieuw. De enige restrictie in het spel is, dat je niet door de muren heen kan gaan.

Er zijn verschillende manieren om het een stukje gemakkelijker maken. Eet "Wasabi" en je zult tijdelijk de kracht weten te vinden om de samurai te verslaan . . . Je zult namelijk in brand staan en bij aanraking met een samurai zul je ze verslaan. Als je genoeg sushi eet zul je beschikken over verschillende "power up's". Hoe meer je eet, hoe beter, aangezien je "powers" steeds krachtiger worden. Het verzamelen van je energie kun je volgen in de vorm van potions. Is het eerste flesje gevuld, dan kun je kiezen om hem te gebruiken of om door te sparen om een betere power op een later moment te kunnen gebruiken. De eerste en tweede power hebben met elkaar te maken. Het zijn namelijk bommen die je kunt neerleggen, met het subtiele verschil dat de tweede bom krachtiger is en dus een groter bereik heeft. De bom explodeert als er een timer is afgelopen of als iemand er overheen loopt. Dit wil dus zeggen dat de speler ook door zijn eigen bom kan worden verslagen. Met de derde power kun je het speelveld doen laten schokken door een sprong te maken. Het gevolg hiervan is dat de samurai tijdelijk gedesoriënteerd zijn en stil blijven staan. Maar pas op! Als je een power gebruikt, maak je een hoeveelheid potions op, afhankelijk van welke power je gebruikt. Je krijgt meer punten voor het verslaan van de samurai (de hoeveelheid samurai is afhankelijk van de grootte van het speelveld) dan door het eten van de sushi, maak hier dus dankbaar gebruik van. Eet ze!

Hoofdstuk 4

Inventarisatie

Voor onze inventarisatie onderscheiden we de volgende 3 elementen: Het spel, het speelveld en de objecten. Deze worden hieronder nader verklaard.

4.1 Spel

Het **spel** is het hoofdelement van het spel, de rest is hieraan gekoppeld.
Eigenschappen:

1. een rij levels, die wij **speelveld**en noemen
2. een **taaru**
3. een **index** voor het huidige **speelveld**

Acties:

1. kan **index verhogen**
2. kan het spel **starten**
3. kan het spel **stoppen**

4.2 Speelveld

Het speelveld bestaat uit een rechthoekig begrensd gelijkmatig verdeeld raster. Ieder vlak uit dat raster geeft een x- en y-coördinaat aan. Objecten zijn door middel van deze x- en y-coördinaten gepositioneerd op dit raster.
Eigenschappen:

1. **hoogte** en **breedte** van het veld (zijn naar onder begrensd, eindig en zijn altijd oneven)
2. heeft speelveld**coördinaten** met een onderlimiet linksonder, oplopend tot (breedte, hoogte) rechtsboven

3. bevat **muren**
4. bevat een **looppad**
5. bevat een **startpositie**

4.3 Objecten

4.3.1 Object

Een **object** heeft bepaalde eigenschappen.
Eigenschappen:

1. Een **object** heeft een **positie** op het looppad.

4.3.2 Bewegend object

Een **bewegend object** erft alle eigenschappen van een **object**.
Extra eigenschappen:

1. heeft een **snelheid**
2. heeft een **richting**

Extra acties:

1. kan van **positie** veranderen
2. kan van **snelheid** veranderen
3. kan van **richting** veranderen

Taaru

Het object **taaru** erft alle eigenschappen en acties van een **bewegend object**.
Eigenschappen:

1. heeft een toestand: **flaming**
2. heeft een **flamingduratie**
3. heeft een **score**
4. heeft een teller voor het aantal gegeten **sushi**'s: **sushi_teller**
5. een teller voor het aantal **potions**: **potion_teller**
6. heeft een teller voor het aantal **levens**: **levens_teller**

Acties:

1. kan **sushi_teller** veranderen
2. kan **potion_teller** veranderen
3. kan **score** veranderen
4. kan **flaming** veranderen
5. kan **flaming_duratie** veranderen
6. kan **levens_teller** veranderen

Samurai

Samurai erven alle eigenschappen en acties van een **bewegend object**.
Eigenschappen:

1. heeft een teller voor het stilstaan: **stilstaan_duur**

Acties:

1. kan **verdwijnen** van het spel
2. kan **stilstaan_duur** veranderen

4.3.3 Eten

Eten erft alle eigenschappen van een **object**.
Eigenschappen:

1. is een sushi of een wasabi

Acties:

1. kan **verdwijnen**

4.3.4 Bommen

Bommen erven alle eigenschappen een **object**.
Eigenschappen:

1. heeft een teller voor het afgaan: **delay_teller**
2. is een kleine_bom of een grote_bom

Acties:

1. kan **delay_teller** veranderen
2. kan **verschijnen**
3. kan **verdwijnen**

4.3.5 Explosie

Explosie erft alle eigenschappen van **object**. Eigenschappen:

1. heeft een explosie duratie: **explosie_duur**

Acties:

1. kan **verschijnen**
2. kan **verdwijnen**
3. kan **explosie_duur** veranderen

4.3.6 Barak

De **barak** erft alle eigenschappen van een **object**.
Eigenschappen:

1. heeft een **teller**

Acties:

1. kan **teller** veranderen

4.4 Flow van het spel

4.4.1 Initialisatie

Voor het spel kan beginnen moet er een speelveld worden gemaakt. De Initialistie wordt gestart en genereerd een speelveld met daarop een looppad. Alles wat geen looppad is, is een muur. Het looppad is hierbij dus nooit doodlopend en er kunnen geen "eilanden" ontstaan. Dit wil zeggen dat delen van het speelveld niet afgesloten kunnen zijn voor "Taaru no Kizoku". Een barak wordt geplaatst in het midden van het veld. Alle overige coördinaten van het looppad worden nu gevuld met sushi of wasabi.

4.4.2 Spel

Het speelveld is nu gemaakt en het spel kan gaan beginnen. "Taaru no Kizoku" krijgt een startpositie en er worden samurai gecreëerd vanuit hun startposite, de barak. De hoeveelheid samurai is afhankelijk van de grootte van het speelveld. Taaru no Kizoku krijgt een bepaald aantal levens die globaal over het spel gelden. Deze levens kunnen verloren gaan op verschillende manieren. Als Taaru dood gaat, zal het aantal levens afnemen tot een ondergrens is bereikt, met als gevolg dat het spel eindigt en er een score zal worden weergegeven.

Door middel van de gebruiker kan Taaru no Kizoku zich verplaatsen over alle coördinaten van het looppad. Taaru kan zich alleen verplaatsen naar een buur-coördinaat op het looppad vanaf zijn huidige positie. Door sushi of wasabi te eten (Taaru bevindt zich op dezelfde

positie als sushi of wasabi), wordt de sushiteller verhoogd. De waarde van de sushiteller komt overeen met een aantal potions. Deze potions kunnen worden gebruikt ten koste van het uiteindelijk te halen punten.

Op ieder gegeven moment kunnen de verkregen potions worden gebruikt in ruil voor Taaru no Kizoku's powers. De powers zullen een verschillend aantal potions kosten. Een mogelijkheid is om een bom neer te leggen, waarvan ook een grotere variant beschikbaar is. Bij het neerleggen van de bom zal er op hetzelfde moment een aflopende timer gaan lopen. Op het moment dat de timer eindigt, zal deze verdwijnen en er zullen er verschillende explosies plaatsvinden waarvan het aantal en de posities afhankelijk zijn van de grootte van de neergelegde bom. Ieder bewegend object dat zich in het bereik van de ontploffing bevindt, zal dood gaan. De bom komt ook tot ontploffing wanneer er een bewegend object zich op dezelfde plek bevindt als de neergelegde bom, waarbij de timer automatisch verdwijnt. Een andere power van Taaru no Kizoku is het laten ontstaan van een aardbeving. Deze zal direct plaatsvinden bij activering en zal ervoor zorgen dat de samurai stil zullen staan. Op dit moment zal er voor alle aanwezige samurai een aflopende timer starten. Wanneer de ondergrens van de timer is bereikt, gaan de samurai weer bewegen. Voor samurai gelden dezelfde regels van verplaatsing als voor Taaru no Kizoku zelf. In het geval dat Taaru no Kizoku een positie deelt met een samurai, zal Taaru no Kizoku dood gaan, waarna het level volledig opnieuw wordt gestart. Wanneer Taaru no Kizoku een Wasabi eet, zal zijn staat veranderen in "flaming". Dit heeft hetzelfde gevolg als de power aardbeving wordt gebruikt, echter wanneer Taaru no Kizoku nu een samurai tegenkomt, zal de samurai dood gaan en zal Taaru no Kizoku een bepaald aantal bonus punten krijgen. Wanneer de stilstaan-timer van de samurai eindigt, zal Taaru no Kizoku zijn toestand terug verranderen naar normaal. De samurai verdwijnt van zijn huidige positie en verschijnt na enige tijd weer op zijn startpositie, de barak.

Wanneer alle sushi en wasabi uit het level zijn verdwenen, wordt de score bijgewerkt op het huidige aantal sushi's en begint een nieuw level door door de index van speelveld te verhogen. Het aantal levens en de score blijven behouden, de andere eigenschappen van pacman worden teruggezet naar hun initiele toestand.

Hoofdstuk 5

Specificatie

Hieronder staat een lijstje van de verzamelingen waar wij gebruik van maken. Later wordt uitgelegd hoe de verzameling is gespecificeerd.

taaru
samurais
sushi
wasabi
kleine_bommen
grote_bommen
speelveldcoördinaten
muren
looppad
barak
bewegend
eten
bommen
spel
objecten
explosies

Wij hebben gebruik gemaakt van de operator $\oplus$ om het variabel zijn van een object aan te duiden. Deze defineren wij als volgt:

object' = object $\oplus$ (gegeven_object, nieuwe_waarde)
waarbij object' op een gekozen moment kan veranderen naar nieuwe_waarde
of, bij een verzameling:
$objectverzameling' = objectverzameling \oplus$ (gegeven_object, nieuwe_waarde)
waarbij het object aangegeven door gegeven_object uit de verzameling *objectverzameling* nieuwe_waarde krijgt, dit maakt dus de *objectverzameling* dynamisch

In principe is dit een flexibele assignment, met het subtiele verschil dat de verandering plaats kán vinden, en dit niet onmiddelijk doet.

5.1 Spel

$spel = \{speelveld, taaru, index\}$

1. $index \in \mathbb{N}$
2. $index\_verhogen : \mathbb{N} \to \mathbb{N}$
   $index\_verhogen(index) = (index')$
   $index' = index \oplus (index, i)$, met $i \in \mathbb{N}$
3. $starten : begin \to spel$
4. $stoppen : spel \to score\_teller$

5.2 Speelveld

1. hoogte, breedte $\in \{x \in \mathbb{N} : x \geq 3 \wedge x \textbf{ mod } 2 = 1 : x\}$
2. $speelveldcoördinaten = \{(x, y) \in \mathbb{N}^2 : x \leq \text{breedte} \ \wedge y \leq \text{hoogte} : (x, y)\}$
3. $buur : \mathbb{N} \times \mathbb{N} \sim \mathbb{N} \times \mathbb{N}$
   $(x_1, y_1) buur (x_2, y_2) \Leftrightarrow (x_1 - x_2)^2 + (y_1 - y_2)^2 = 1$
4. $looppad = \{(x_1, y_1) | \exists_{x_2, y_2} [(x_2, y_2) \in looppad | \exists_n [n : n \in \mathbb{N}^+ : (x_1, y_1) buur^n (x_2, y_2)]]\}$
5. $muren = speelveldcoördinaten \setminus looppad$
6. startpositie $\in looppad$

5.3 Objecten

1. $objecten = bewegend \cup eten \cup bommen \cup explosies \cup barak$

In de volgende specificaties wordt bij bepaalde functies gebruik gemaakt van ¡object¿.¡eigenschap¿, waarbij ¡object¿ is meegegeven bij de functie en ¡eigenschap¿ deel uitmaakt van de gespecificeerde tupels van het ¡object¿. Bijvoorbeeld positie.x in de eerstvolgende functie. Positie is meegegeven en bestaat uit een x- en y-coördinaat die door positie.x en positie.y kunnen worden gebruikt.

5.3.1 Bewegend object

bewegend = { (x, y) | x = (bewegend ∨ stilstaand) ∧
y = (noord ∨ oost ∨ zuid ∨ west)}

loop_naar_noord: *looppad* → *looppad*
loop_naar_noord(positie) = (positie')
positie' = positie ⊕ (positie, (x,y)), met:
(x,y) = (positie.x, (positie.y)+1) als (positie.x, (positie.y)+1) ∈ *looppad* en anders (x, y) = (positie.x, positie.y)

loop_naar_oost: *looppad* → *looppad*
loop_naar_oost(positie) = (positie')
positie' = positie ⊕ (positie, (x,y)), met:
(x,y) = ((positie.x)+1, positie.y) als ((positie.x)+1, positie.y) ∈ *looppad* en anders (x, y) = (positie.x, positie.y)

loop_naar_zuid: *looppad* → *looppad*
loop_naar_zuid(positie) = (positie')
positie' = positie ⊕ (positie, (x,y)), met:
(x,y) = (positie.x, (positie.y)-1) als (positie.x, (positie.y)-1) ∈ *looppad* en anders (x, y) = (positie.x, positie.y)

loop_naar_west: *looppad* → *looppad*
loop_naar_west(positie) = (positie')
positie' = positie ⊕ (positie, (x,y)), met:
(x,y) = ((positie.x)-1, positie.y) als ((positie.x)-1, positie.y) ∈ *looppad* en anders (x, y) = (positie.x, positie.y)

Taaru

taaru = { (x, positie, flaming, flaming_duratie, score_teller, sushi_teller, potion_teller, levens_teller) |$x \in$ *bewegend* $\wedge$ *positie* $\in$ *looppad* $\wedge$
$flaming \in \mathbb{B} \wedge flaming\_duratie \in \mathbb{N} \wedge$
$score\_teller \in \mathbb{N} \wedge sushi\_teller \in \mathbb{N} \wedge$
$potion\_teller \in \mathbb{N} \wedge levens\_teller \in \mathbb{N}$}

Acties:

1. verander flaming: $\mathbb{P}(taaru) \times \mathbb{B} \rightarrow \mathbb{P}(taaru)$
   verander flaming(*taaru*, bool) = (*taaru*')
   taaru' = *taaru* ⊕ (*taaru*, (taaru.x, taaru.positie, bool, taaru.flaming_duratie, ..., taaru.levens_teller)), met *bool* $\in \mathbb{B}$

2. verander flaming_duratie: $\mathbb{P}$(taaru) $\times \mathbb{N} \rightarrow \mathbb{P}$(taaru)
   verander flaming_duratie(*taaru*, duratie) = (*taaru*')
   taaru' = *taaru* $\oplus$ (*taaru*, (taaru.x, taaru.positie, taaru,flaming, duratie, taaru.score_teller, ..., taaru.levens_teller)), met *duratie* $\in \mathbb{N}$

3. verander score_teller: $\mathbb{P}$(*taaru*) $\times \mathbb{N} \rightarrow \mathbb{P}$(*taaru*)
   verander score_teller(*taaru*, score) = (*taaru*')
   taaru' = *taaru* $\oplus$ (*taaru*, (taaru.x, ..., taaru.duratie, score, taaru.sushi_teller ..., taaru.levens_teller)), met *score* $\in \mathbb{N}$

4. verander sushi_teller: $\mathbb{P}$(*taaru*) $\times \mathbb{N} \rightarrow \mathbb{P}$(*taaru*)
   verander sushi_teller(*taaru*, sushi) = (*taaru*')
   taaru' = *taaru* $\oplus$ (*taaru*, (taaru.x, ..., taaru.score, sushi, taaru.potion_teller, taaru.levens_teller)), met *sushi* $\in \mathbb{N}$

5. verander potion_teller: $\mathbb{P}$(*taaru*) $\times \mathbb{N} \rightarrow \mathbb{P}$(*taaru*)
   verander potion_teller(*taaru*, potion) = (*taaru*')
   taaru' = *taaru* $\oplus$ (*taaru*, (taaru.x, ..., taaru.sushi_teller, potion, taaru.levens_teller)), met *potion* $\in \mathbb{N}$

6. verander levens_teller: $\mathbb{P}$(*taaru*) $\times \mathbb{N} \rightarrow \mathbb{P}$(*taaru*)
   verander levens_teller(*taaru*, levens) = (*taaru*')
   taaru' = *taaru* $\oplus$ (*taaru*, (taaru.x, ..., taaru.potion_teller, levens)), met *levens* $\in \mathbb{N}$

Samurais

samurais = { (x, positie, stilstaan_duur) | (x $\in$ *bewegend*) $\wedge$ (positie $\in$ *looppad*) $\wedge$ (stilstaan_duur $\in \mathbb{N}$) }

Acties:

1. verdwijn: $\mathbb{P}$(*samurais*) $\times$ *looppad* $\rightarrow \mathbb{P}$(*samurais*')
   verdwijn (*samurais*, positie') = (*samurais*')
   samurais' = *samurais* $\setminus$ {x, positie, stilstaan_duur | positie = positie'}

2. verander stilstaan_duur: $\mathbb{P}$(samurais) $\times \mathbb{N} \rightarrow \mathbb{P}$(samurais)
   verander stilstaan_duur (*samurais*, nieuwe_duur) = (samurais')
   samurais' = stilstaan_duur $\oplus$ (samurais, (samurais.x, samurais, positie, nieuwe_duur)

5.3.2 Eten

$eten$ = { (soort, positie) | soort = (sushi $\vee$ wasabi) $\wedge$ positie $\in$ *looppad* }

Acties:

1. verdwijn: $\mathbb{P}$(eten) $\times$ *looppad* $\rightarrow$ $\mathbb{P}$(eten)
   verdwijn (*eten*, eten_positie) = (*eten*')
   eten' = *eten* $\oplus$ (eten_positie, $\emptyset$)

$sushi$ = { (soort, positie) $\in$ *eten*| (soort = sushi) }
$wasabi$ = { (soort, positie) $\in$ *eten*| (soort = wasabi) }

5.3.3 Bommen

$bommen$ = { (soort, delay_teller, positie) | soort = (klein $\vee$ groot) $\wedge$ delay_teller $\in \mathbb{N} \wedge$ positie $\in$ *looppad* }
$kleine\_bommen$ = { (soort, delay_teller, positie) $\in$ *bommen*| (soort = *klein*) }
$grote\_bommen$ = { (soort, delay_teller, positie) $\in$ *bommen*| (soort = *groot*) }

Acties:

1. verander delay_teller: $\mathbb{P}(bommen) \times \mathbb{N} \rightarrow \mathbb{N}$
   verander delay_teller(*bommen*, i) = (*bommen*')
   $bommen' = bommen \oplus$ (*bommen*, (bommen.soort, i, bommen.positie)), met $i \in \mathbb{N}$

2. verschijn: $\mathbb{P}(bommen) \times looppad \rightarrow \mathbb{P}(bommen)$
   verschijn(*bommen*, positie) = ($bommen'$)
   $bommen' = bommen \oplus (bommen,(bommen.soort, bommen.delay_teller,$positie)

3. verdwijn: $\mathbb{P}(bommen) \rightarrow \mathbb{P}(bommen)$
   verdwijn(*bommen* = ($bommen'$)
   $bommen' = bommen \oplus$ (*bommen*, (bommen.soort, bommen.delay_teller, $\emptyset$))

5.3.4 Explosies

$explosies$ = { (positie, duur) | (positie $\in$ *looppad*)$\wedge$ (duur $\in \mathbb{N}$) }
Acties:

1. verschijn: *looppad* $\times \mathbb{P}$(explosies) $\rightarrow \mathbb{P}$(explosies)
   verschijn(explosie.positie, $explosies$) = ($explosies'$)
   $explosies' = explosies \oplus (explosies, explosies \cup \{$(explosie.positie,i)$\})$, met $i \in \mathbb{N}$

2. verdwijn: $looppad \times \mathbb{P}(\text{explosies}) \rightarrow \mathbb{P}(\text{explosies})$
   verdwijn(explosie.positie, $explosies$) = ($explosies'$)
   $explosies' = explosies \oplus (explosies, explosies \setminus \{(\text{explosie\_positie,i})\})$, met $i \in \mathbb{N}$

3. verander duur: $\mathbb{P}(explosies) \rightarrow \mathbb{P}(explosies)$
   verander duur(explosie) = (explosie')
   explosie' = explosie $\oplus$ (explosie, (explosie.positie, i)), met $i \in \mathbb{N}$

5.3.5 Barak

$barak = \{$ (positie, teller) | positie $\in looppad \wedge$ teller $\in \mathbb{N}$ $\}$

1. verander teller: $barak \rightarrow barak$
   verander teller(barak, i) = (barak')
   barak' = teller $\oplus$ (barak, (barak.positie, i)), met $i \in \mathbb{N}$

Hoofdstuk 6

Synthese

Voor onze opdracht werd ons niet verplicht gesteld om de synthese volledig uit te werken. Wél werd ons gevraagd om hierover na te denken.

Ons lijkt het zeker mogelijk om de hele samenhang van het spel formeel te defineren in een flowchart. Dit werkt goed samen met de acties op objecten zoals wij die nu hebben geimplementeerd.

Een voorbeeld zou een begin en einde bevatten waartussen een initialisatie voor het spel bevind, waarna een loop plaatsvindt waarin een initialisatie van het speelveld plaatsvindt, samen met het verdere verloop van het spel. Het einde bijvoorbeeld is dan direct gekoppelt aan de situatie waarin het aantal levens 0 is en Taaru een leven verliest.