# Module User Experience | Gaming

## Onderdeel Gameplay Activiteit 2: Mechanics en regels

## Auteurs

Paul Bergervoet (Gameplay en teamleiding)

Monique Dewanchand (C#)

Lars Tijsma (Unity)

Anton Visser (Gameplay en testen)

# 

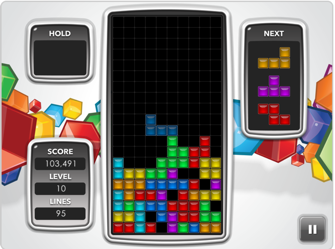
# 

# Activiteit 2: Mechanics, de basis

## -- Mechanics: het spel zelf

In de inleiding van deze module hebben we al kort gekeken naar de mechanics, de werking van een spel. We herhalen hier het stukje over de mechanics van Tetris.

Bij het spel *Tetris* heb je blokjes die opgebouwd zijn uit vier vierkantjes. Die blokjes vallen naar beneden in het verder lege speelveld. Als speler kun je de blokjes naar links en naar rechts schuiven en je kunt ze draaien, met de pijltoetsen. Op de bodem van het speelveld moet je de blokjes zo in elkaar passen dat er volle regels ontstaan. Die verdwijnen dan en daarvoor krijg je punten. Doel is om zo veel mogelijk punten te halen voor je af gaat, omdat het speelveld helemaal vol met blokjes is. De blokjes vallen steeds sneller, dus er komt een moment dat je het niet meer bijhoudt...



*Tetris, te spelen via tetris.com/play-tetris*

Tetris is natuurlijk een eenvoudig - maar goed bedacht - spel, waarin alles bepaald wordt door de zes soorten blokken die kunnen voorkomen. Daardoor heeft het spel een heel eigen mechanic.

Voor de meeste spellen is dat niet zo. Veel spellen hebben mechanics die op elkaar lijken, met variaties. Je kunt spreken van *soorten games*. In de inleiding van de module noemden we er al een paar, hier gaan we daarop door.

* **Platform Game**: je beweegt horizontaal (meestal naar rechts) en springt om op platforms te komen. Daarbij moet je obstakels en vijanden vermijden en andere objecten juist oppakken.
* **Candy Crush / Bejeweled**: in een bord vol stenen, vruchten, snoepjes of … maak je rijtjes van gelijke stenen of snoepjes door twee naast elkaar liggende dingen te verwisselen.
* **First Person Shooter**: Je speelt een soldaat of krijger die door een gevaarlijke wereld loopt waar de hele tijd tegenstanders opduiken. Schiet de tegenstanders neer!
* **Adventure** of **Role Playing Game**: je beweegt in een grote wereld waar van alles te vinden is. Je verzamelt objecten, krijgt opdrachten of puzzels die je moet oplossen
* **Sim Game**: bouw een sociaal netwerk, een stad of zelfs een wereldrijk door hulpbronnen (resources) te vinden en slim in te zetten.
* **Race Game** of **Parcours Game**: leg een parcours af met een of ander vervoermiddel (van auto tot ski’s), race tegen anderen.
* Enz. enz. enz...

De soorten games hierboven hebben steeds een bepaalde mechanic die het belangrijkst is, bijvoorbeeld rondlopen en schieten bij een First Person Shooter. Dat noemen we de **primary mechanic**.

Naast de *primary mechanic* hebben grotere spellen gebruiken vaak meer mechanics. In een schietspel is het niet alleen schieten, maar je moet soms ook iets anders: ergens een deur openen bijvoorbeeld. In een adventure game waar je schatten verzamelt, moet je ook wel eens een puzzel oplossen.

Mechanics bepalen de gameplay en daarmee de spelervaring. Het is dus het belangrijkste onderdeel van je game-ontwerp. We gaan er uitgebreid naar kijken. Om te beginnen de basics: speleracties, regels en feedback. Dat bekijken we in deze activiteit.

In de volgende activiteit kijken we naar speciale mogelijkheden om mechanics te maken: variatie door gebruik van objecten (zoals wapens), gebruik maken van tijd, toeval en meer.

## -- Speleracties, regels en feedback

De kern van de gameplay gaat om *acties*, *regels* en *feedback*. Kort gezegd: Als speler *doe* je iets en het spel reageert. Die reactie wordt bepaald door *regels*. Het resultaat krijg je te zien of te horen: *feedback*.

* **Acties die de speler kan doen**  
  Om te beginnen is belangrijk wat de speler allemaal kan doen. In spellen waar je als poppetje / personage rondloopt is dat bijvoorbeeld: lopen, springen, schieten, iets pakken, een actie met een werktuig uitvoeren.  
  In spellen zonder zichtbare speler verander je het spel met toetsen of de muis. Bij Tetris bijvoorbeeld, zijn er vier mogelijke acties:
  + Pijltoetsen links en rechts bewegen het blokje naar links en rechts
  + Pijltoets omhoog draait het blokje
  + Pijltoets naar beneden laat het blokje sneller vallen
* **Regels**   
  Alle acties en de dingen die je doet met de objecten in het spel zijn gebonden aan regels. Dat is meestal veel meer dan je op het eerste gezicht ziet. Neem bijvoorbeeld het bewegen in een platformgame. Hiervoor kun je een hele rij regels opstellen:
  + Hoe snel beweeg je eigenlijk? Is dat altijd hetzelfde, of hangt dat af van de wereld (bijvoorbeeld gladde platforms)?
  + Hoe hoog en hoe snel spring je? Kun je een sprong maken in een sprong? Kun je horizontaal bewegen tijdens een sprong?
  + Zijn er obstakels waar je niet doorheen kunt?
  + Wat gebeurt er als je van een platform afloopt?

Alle antwoorden op deze vragen worden bepaald door regels en worden vastgelegd in stukjes programmacode: in Unity in de scripts. Het goed formuleren van regels is zo belangrijk, dat we er zometeen dieper op ingaan.

* **Feedback**  
  Wanneer je als speler allerlei acties uitvoert, wil je ook weten wat het resultaat is. Het spel moet je feedback geven: Ben je op de goede weg om het doel te bereiken? Heb je weer een stap gezet in *leveling up* van je personage? Is een tegenstander in een gevecht aan het verzwakken?   
  Dat kan op allerlei manieren. Soms zie je de resultaten gewoon in het spel. In First Person Shooters vallen de vijanden om als ze komen te overlijden. In veel spellen staat je score in beeld.  
  Soms zie je ook health-metertjes, die aangeven of een tegenstander - en jij zelf - verzwakken. Soms heb je ook speciale alerts: Unreal heeft een metertje voor je eigen gezondheid, de randen van het scherm kleuren rood als het niet zo goed met je (personage) gaat.



*Combat in Pine*

#### Opdracht GD.4: Acties, regels en feedback in game X

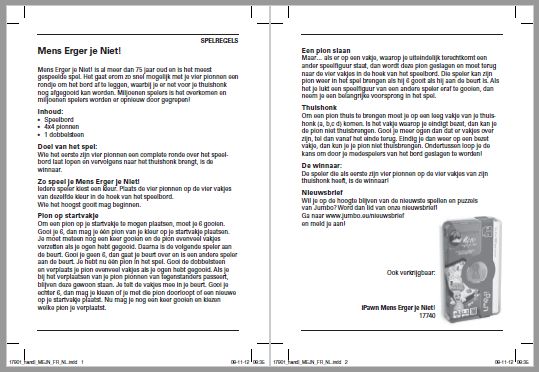
Bij deze module heb je een favoriete game gekozen, voor de analyse van de gameplay. Deze hadden we *game X* genoemd. In game X zal ook sprake zijn van *acties, regels* en feedback. Noem er een paar. Dat kan er zo uitzien:  
*In het spel kan ik actie … doen en het resultaat wordt bepaald door regel … Het resultaat zie ik door …*

Vul de powerpoint-presentatie met de analyse van game X aan.

## -- Spelregels in sport en spel

Niet alleen computerspellen hebben regels, alle spellen hebben *spelregels*, waaraan de spelers zich moeten houden. Die regels zijn soms simpel, maar meestal is het veel meer dan je zou denken. Spelregels moeten namelijk alle mogelijke situaties afdekken, ook de zaken die wat minder vaak voorkomen. Daardoor zijn de spelregels van sommige sporten zware boekwerken, bijvoorbeeld *golf*. Een korte samenvatting van de regels is een [powerpoint-presentatie](https://www.ngf.nl/~/media/pdfs/ngf/regel-en-handicap/clubpresentatie-regels-2019.pdf?rev=1389504875) van 64 slides!

Hieronder zie je een kopie van het blaadje met de officiëlespelregels van Mens-erger-je-niet. Een simpel spel heeft al een behoorlijke lijst regels!



*De regels van Mens-erger-je-niet door fabrikant Jumbo. NB: de afbeelding is zo verkleind dat het onleesbaar is. De precieze regels doen er ook niet toe. Het gaat erom dat zo’n simpel spel al meer dan een pagina aan regels heeft.*

Bij spellen in de echte wereld geven de regels soms gedoe. Je hebt vast wel eens meegemaakt dat er een discussie ontstaat over of iets nu wel of niet mag. Neem bijvoorbeeld voetbal: de discussies over of iets nu wel of niet *hands* is, zijn soms heftig!

Op een spelletjesavond moet je maar zien dat je er samen uitkomt, in competities zijn er scheidsrechters die beslissen.



*Scheidsrechter (umpire) bij honkbal (links in beeld)*

## -- Regels in computerspellen

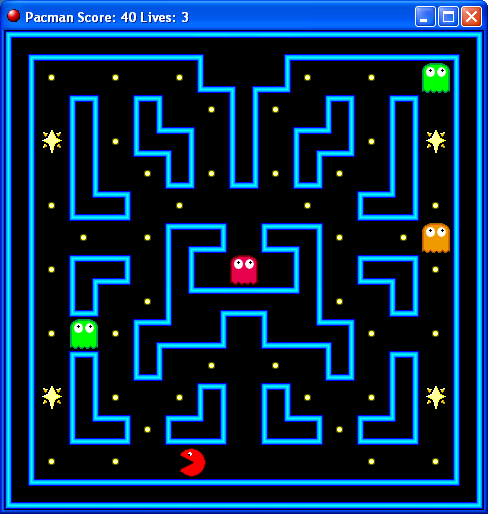
Bij computerspellen zijn er geen discussies, ook geen scheidsrechters. De regels zitten in de programmaregels van het spel en de computer voert ze uit. Dat maakt dingen makkelijk voor de spelers - geen discussie over of iets wel of niet mag.

Voor de makers is het lastig. Je moet alle regels precies formuleren en vertalen in scripts. Een computerspel doet niet zomaar ongeveer iets, de programmaregels zijn precies.

( Of beter gezegd, je kunt een computerspel wel ‘ongeveer’ iets laten doen, maar dat is nog veel lastiger om te programmeren dan een harde regel! Dat is namelijk ‘kunstmatige intelligentie’.)

In deze activiteit gaan we proberen om de regels van een niet al te ingewikkeld spel heel precies te formuleren. Wanneer je de regels precies hebt, kun je er een script van maken.

Dat spel is Pacman. Hieronder zie je een afbeelding.



*Pacman, oudere versie.*

Om de regels te vinden, moet je het spel even kunnen spelen. Er zijn allerlei online versies van Pacman, je kunt bijvoorbeeld de [Pacman van Masswerk](https://www.masswerk.at/JavaPac/JS-PacMan2.html) proberen.

#### Opdracht GD.5: Regels van Pacman

Stel een zo compleet mogelijke lijst van regels van Pacman op,van de versie die je online speelt.  
Houd daarbij rekening met:

* De bewegingen van de speler en andere game-objecten
* De gamewereld waarin het spel zich afspeelt
* Het raken van game-objecten, zowel de bewegende als de stilstaande
* Tijd, snelheid, score, ...

#### Opdracht GD.6: Primary mechanic en regels van je eigen game

Bij de vorige activiteit heb je het concept van je game geformuleerd. Je kunt nu starten met het *design document*. Dat begin je natuurlijk met een inleiding waarin je het concept nog eens herhaalt. Daarna kun je dingen gaan uitwerken.

1. Beschrijf de *primary mechanic* van je game.   
   Maak eventueel gebruik van de lijst eerder in deze activiteit of vergelijk met andere spellen, die je zelf speelt.
2. Schrijf een aantal regels van de primary mechanic op.   
   Wees precies, maar niet al te! Veel instellingen kun je nog niet al te precies maken. Twee of drie vijanden tegelijk? Je weet het nog niet. Zulke instellingen worden pas definitief wanneer je het spel gaat testen.

Gebruik bij deze opdracht bijvoorbeeld de lenzen 31 (*Action*), 33 (*Rules*) en 39 (*Meaningful Choices*) uit de app van Jesse Schell.