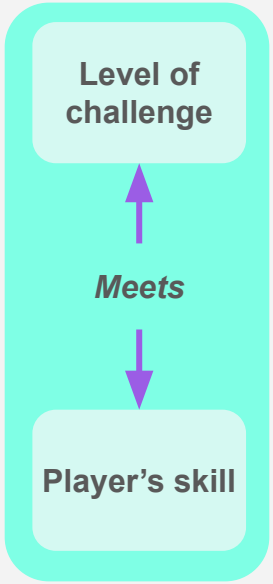
Dynamic Difficulty Adjustment (DDA) as Inspiration for Game Audio

11 Feb. ‘25

Introduction

Dynamic Difficulty Adjustment



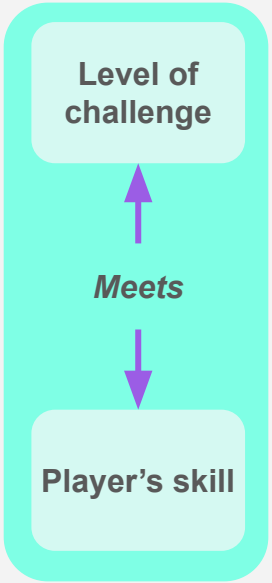
Difficulty

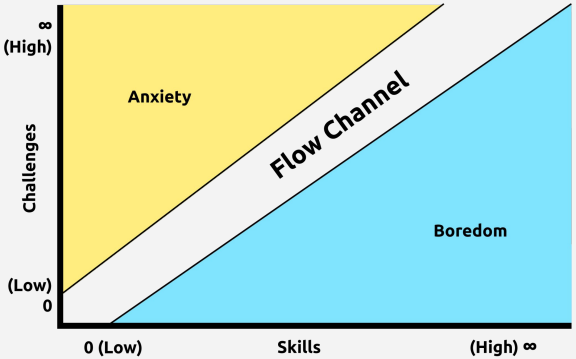
Important aspect of game design.

Matching level of challenge
→ *higher entertainment*

(Sweetser & Wyeth, 2005, Lach, 2017)

Difficulty

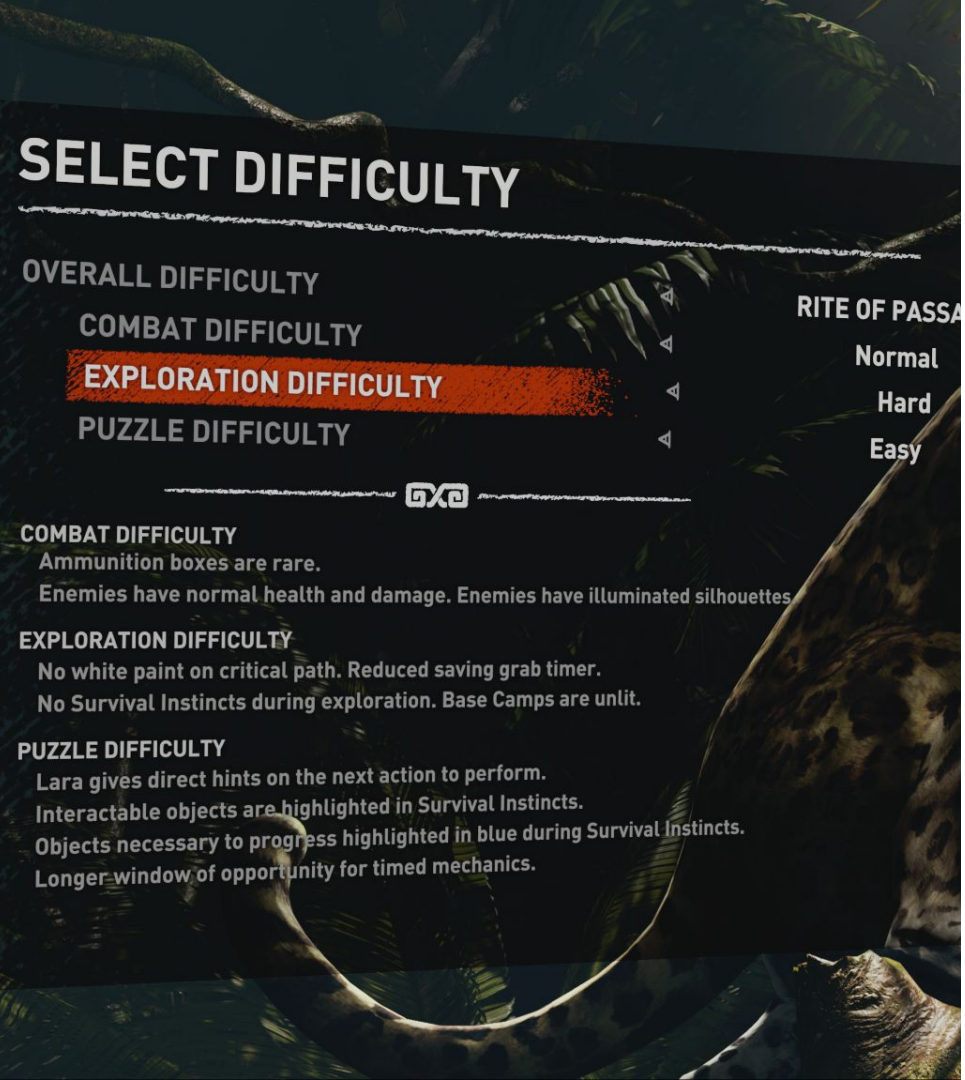




Important aspect of game design.

State of flow = an enjoyable state with peak productivity, complete immersion

(Csikszentmihalyi, 1990, Koster, 2004, Sweetser & Wyeth, 2005, Lach, 2017)

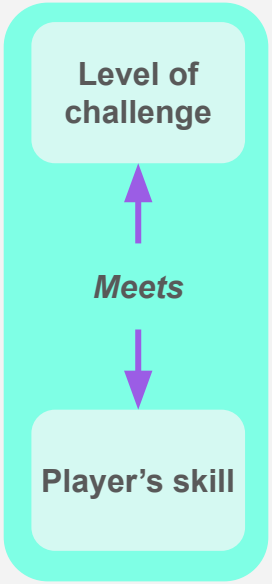


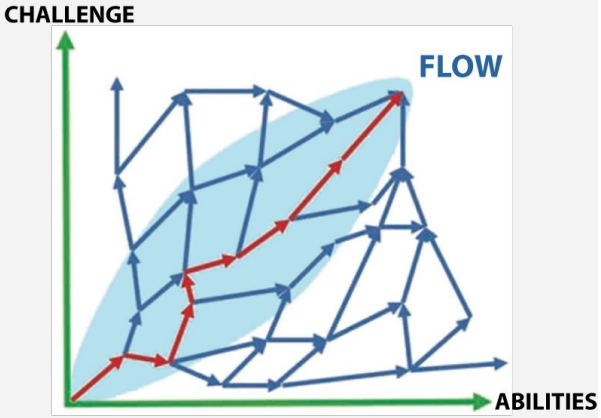
Static difficulty level of games

- problems with **static difficulty** levels
  - too coarse and broad → do not suit the actual level of the player

(2018, Chayer, 2014, Hunicke & Chapman, 2004, Lach, 2017, Glassner, 2001, Andrade et al., 2005)

Difficulty



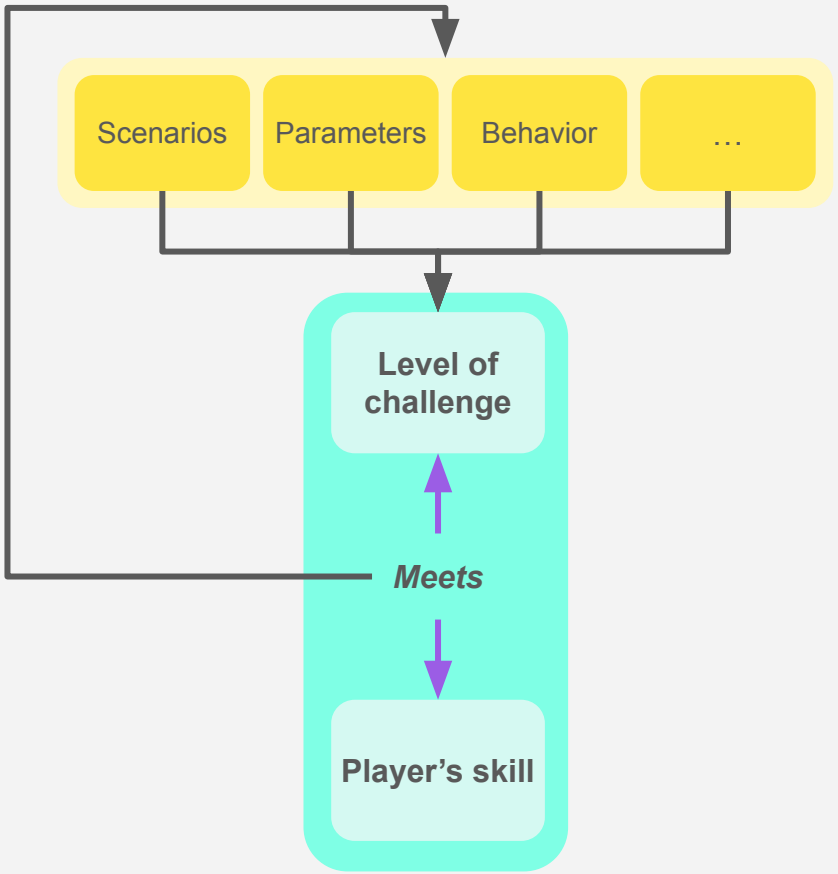


Important aspect of game design.

Adapting to players' flow zones by, e.g.:

- Offering many choices
- Applying dynamic difficulty adjustment (DDA)

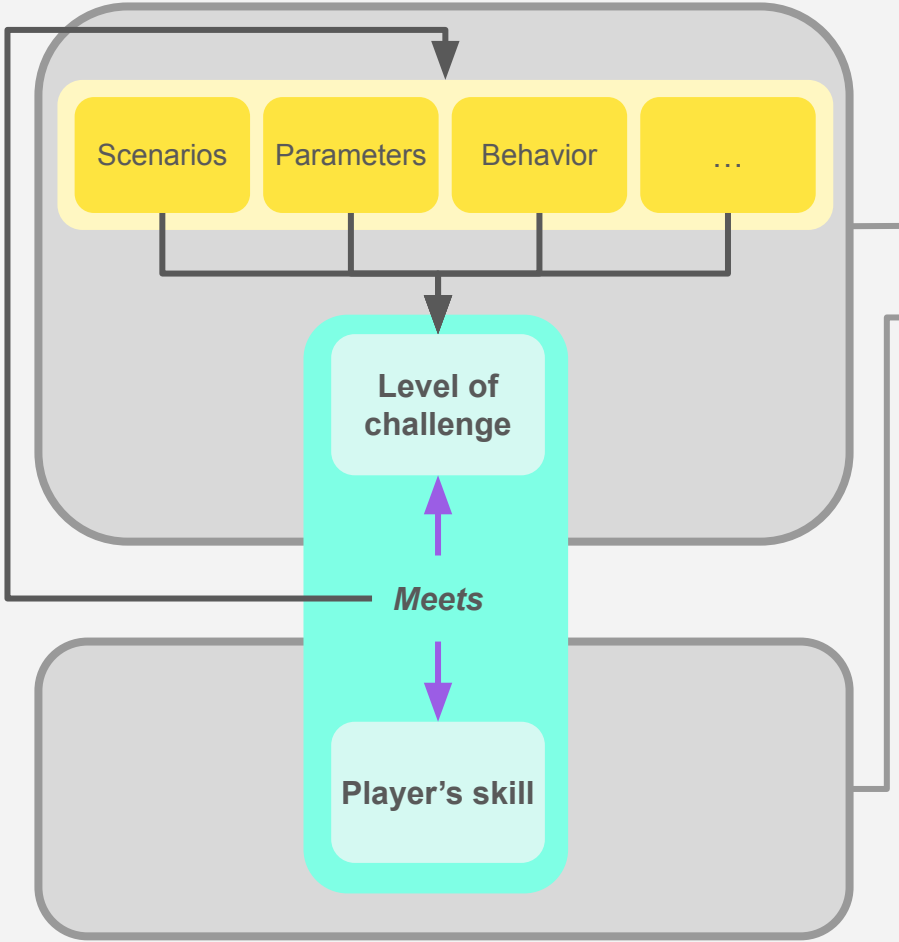
(Koster, 2004, Chen, 2007)



Dynamic Difficulty Adjustment (DDA)

*"... a technique of **automatic real-time adjustment** of scenarios, parameters, and behaviors in video games, which **follows the player's skill** and keeps them from boredom or frustration."*

(Zohaib, 2018)

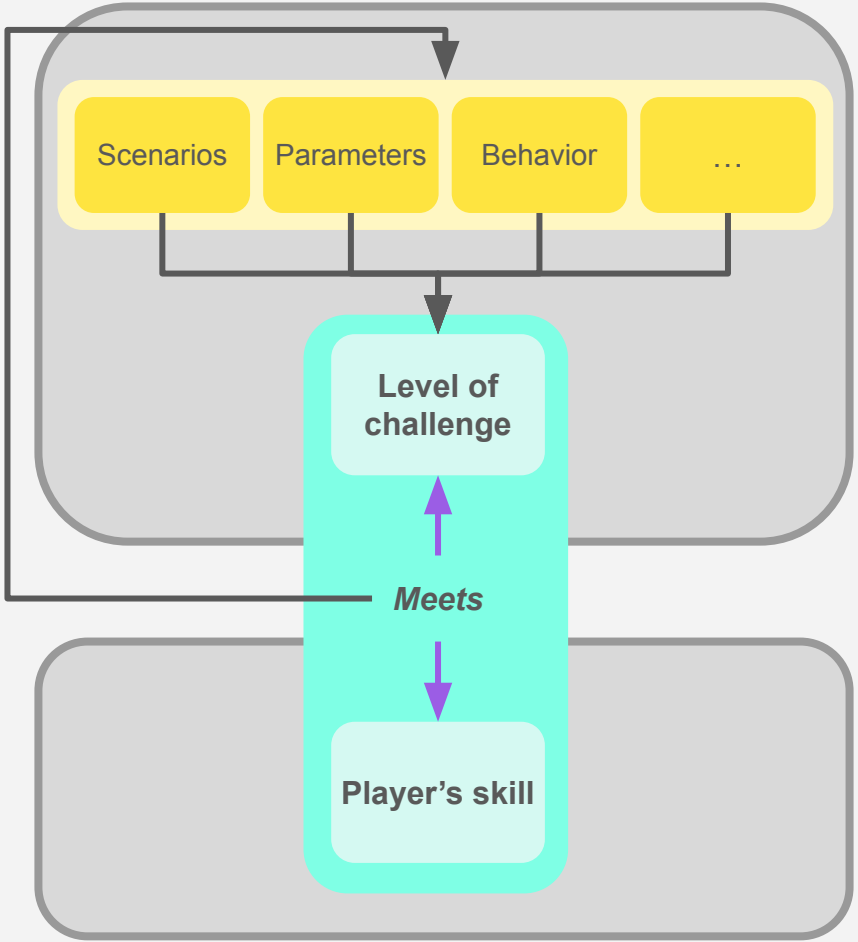


Dynamic Difficulty Adjustment (DDA)

Two main components:

- Difficulty adjustment mechanism
- Player assessment mechanism

(Zohaib, 2018, Adams, 2014)



Dynamic Difficulty Adjustment (DDA)

Example 1

Monitoring game statistics - predefined metrics, e.g. number of failures

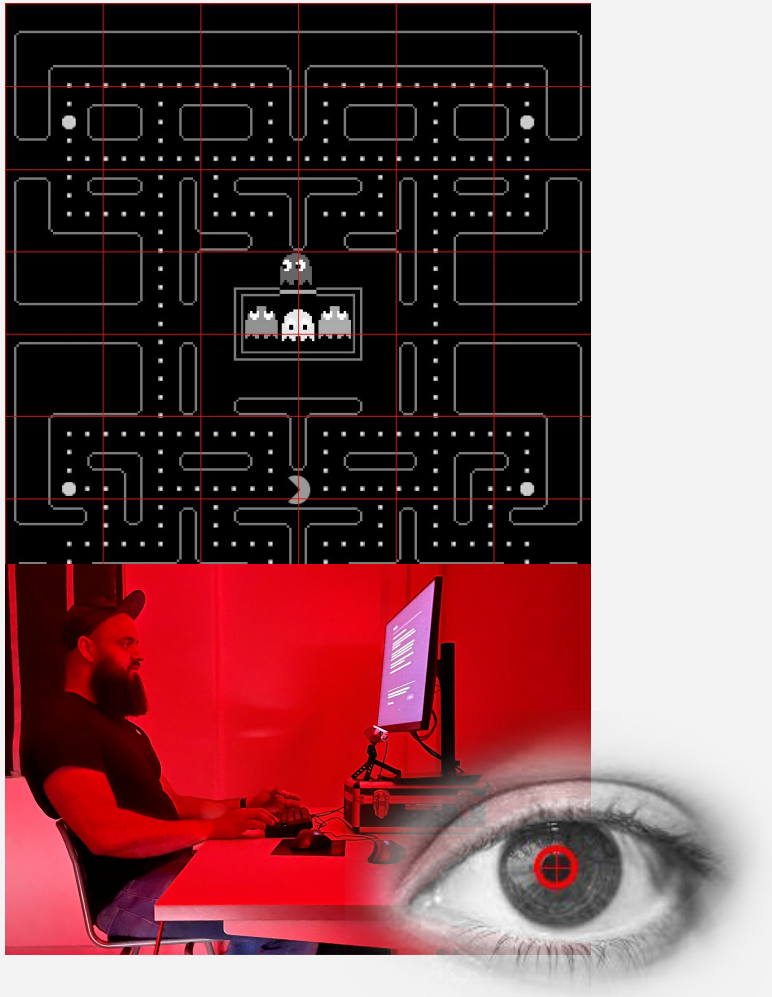
Supply of inventory mechanics, like health, ammunition, shielding, weapons

- # items in the field
- Items & player properties

Demand

- E.g. properties of enemies

(Zohaib, 2018, Adams, 2014, Hunicke & Chapman, 2004)



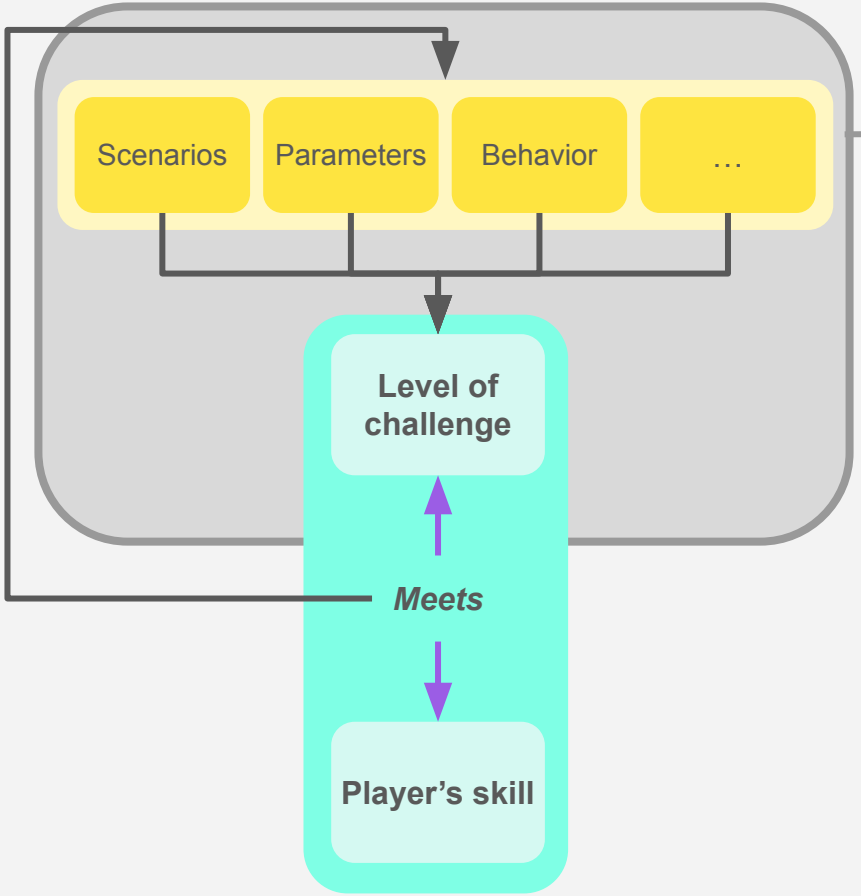
Dynamic Difficulty Adjustment (DDA)

Example 2

Experienced difficulty estimation based on **gameplay features** and **pupillometry measurements**.

(Vriezenga 2024)

Dynamic Difficulty Adjustment (DDA)



Difficulty adjustment mechanism,

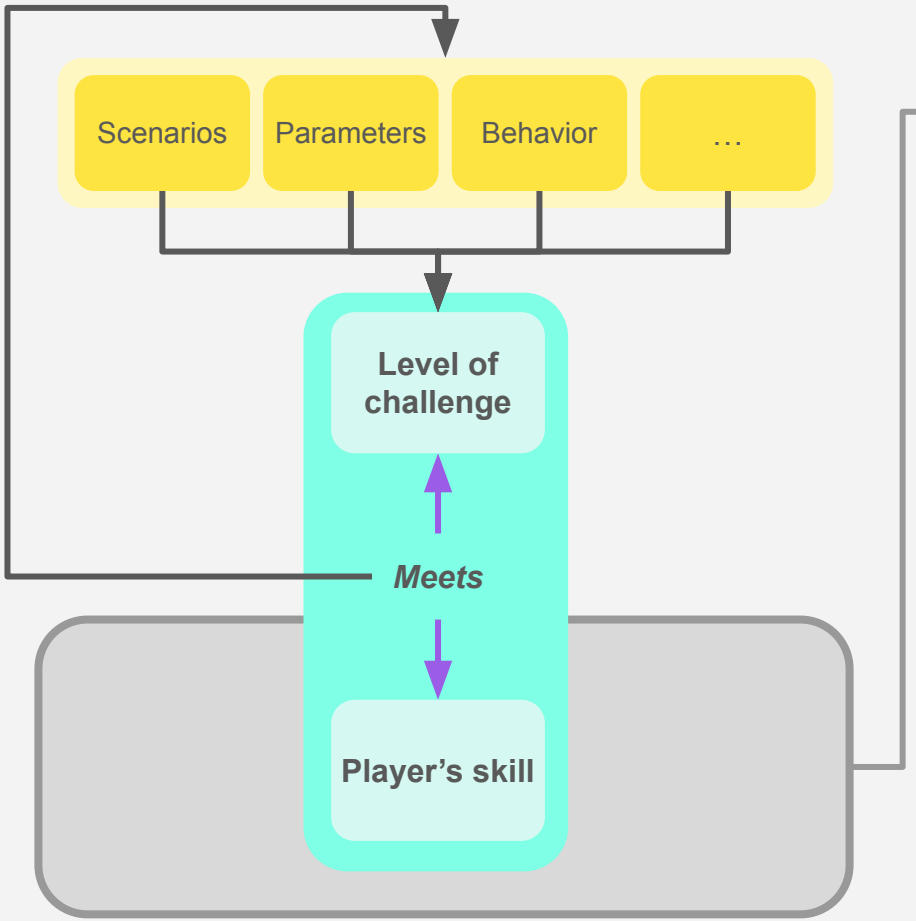
'Adaptive game components'

- Attributes
- Behavior
- Game level/-world layout
- Events
- Scenario and narrative

- Gameplay mechanics

(Hunicke, 2005, Lopes & Bidarra, 2011, Bontchev et al., 2016, Zohaïb, 2018, Sepulveda et al. 2019)

Dynamic Difficulty Adjustment (DDA)



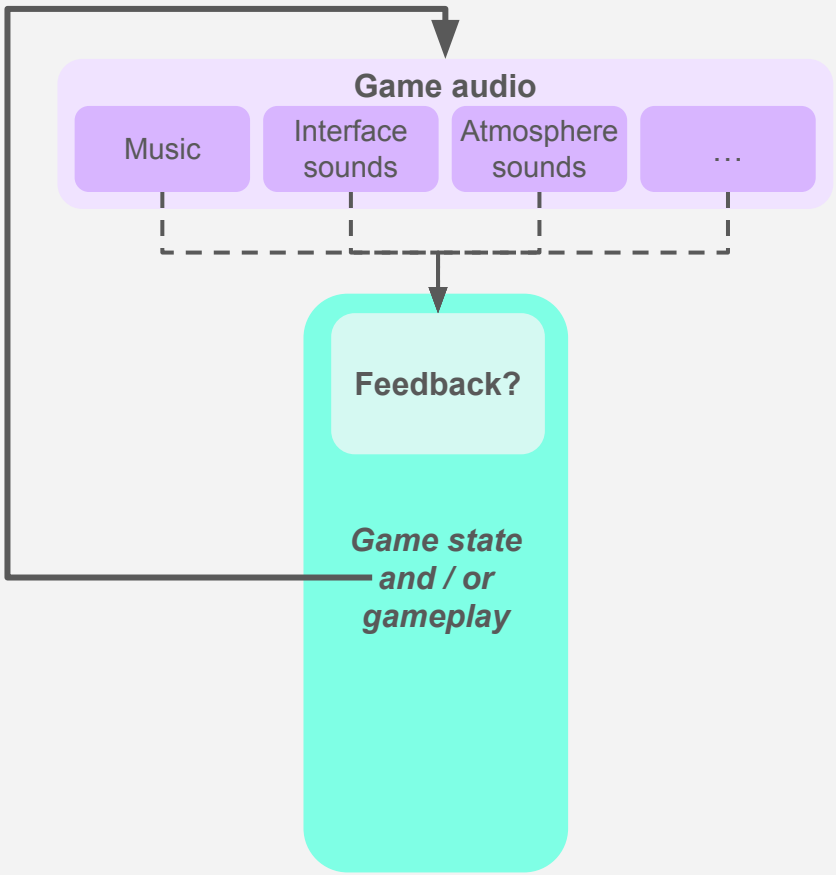
Player assessment mechanism

E.g. player experience modeling

- Gameplay data
- Game context data
- Objective

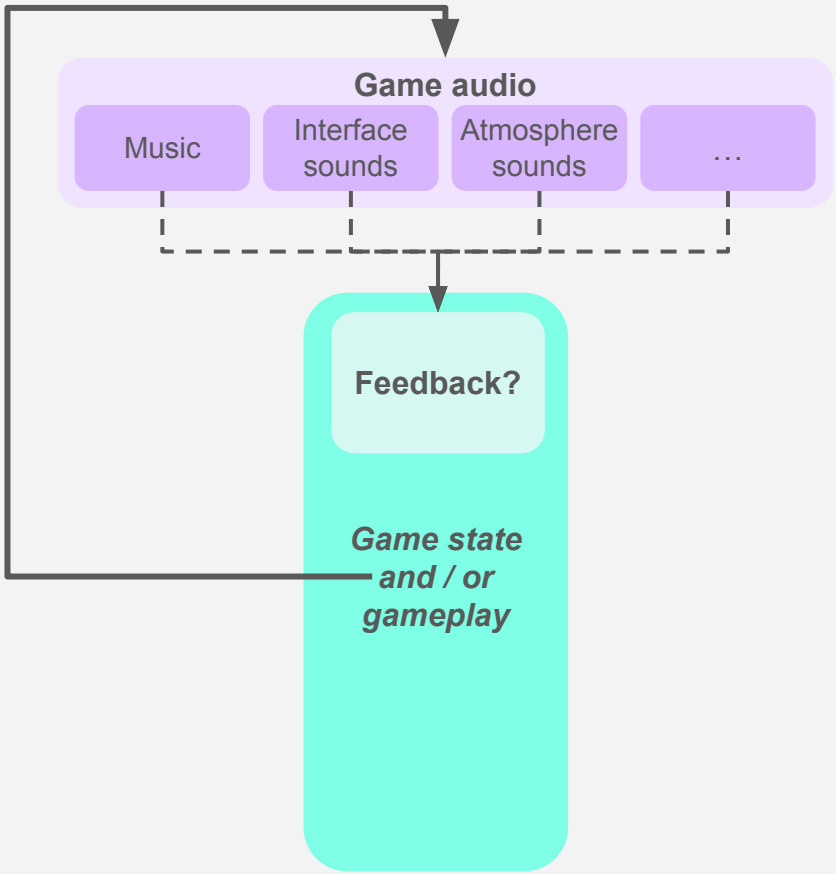
- Player profile
- Linked data

(Lopes & Bidarra, 2011, Yannakakis & Togelius, 2013, Bontchev et al., 2016, , Yannakakis & Togelius, 2018, Zohaïb, 2018, Sepulveda et al. 2019, Paraschos & Koulouriotis, 2022, Lopes & Lopes, 2022)



Overlap with Game Audio

Using a (combined) parameter to alter the game audio



Overlap with Game Audio

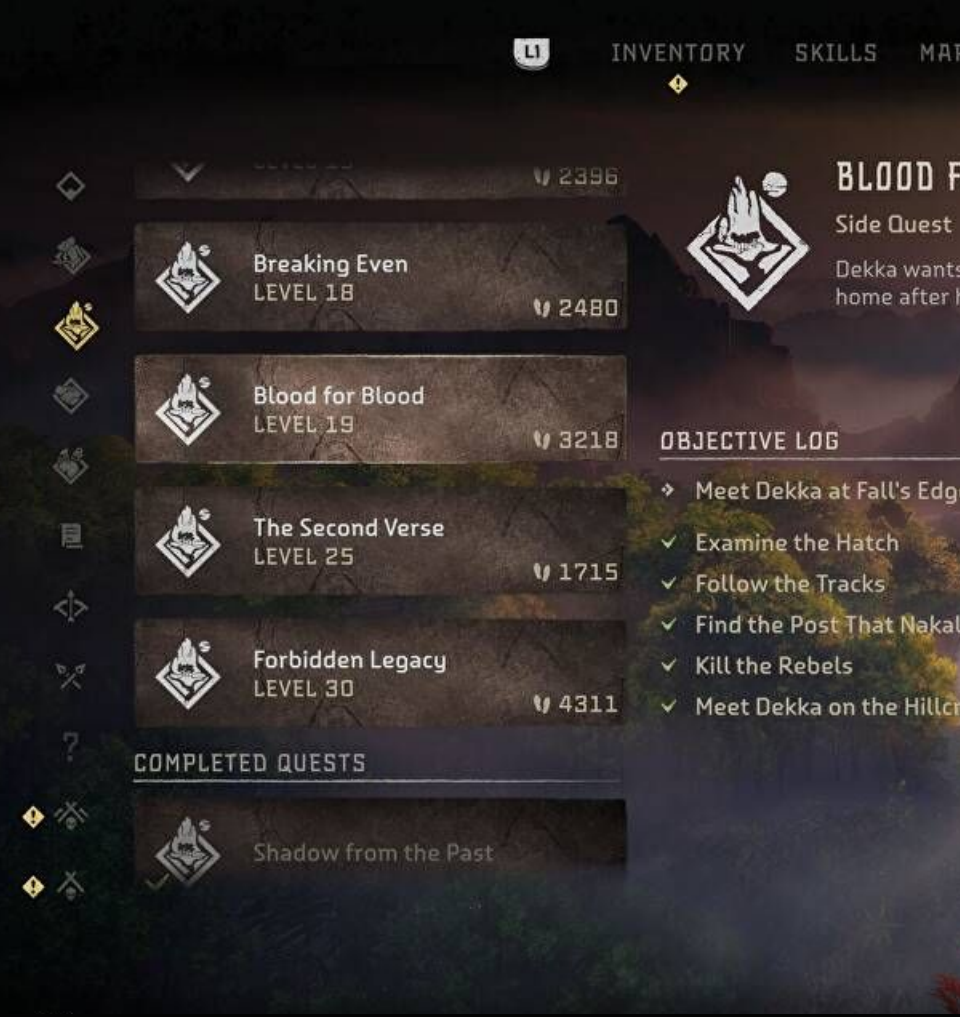
Using a (combined) parameter to alter the game audio.

→ **DDA as inspiration for Game audio**

DDA aspects

Challenge & difficulty

DDA requirements



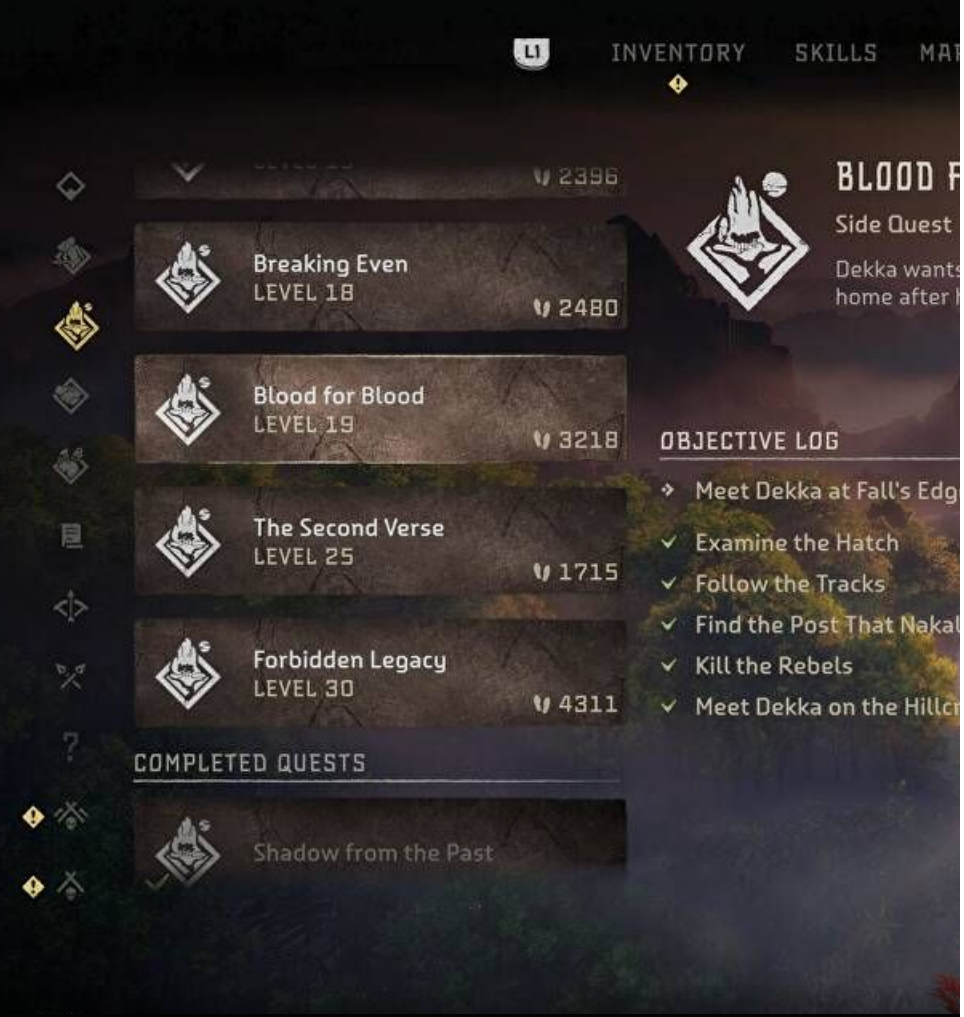
Screenshot from Horizon Zero Dawn (Guerilla Games, 2017)

DDA - Game difficulty

Difficulty types

- Absolute difficulty
  = Required intrinsic skill + Imposed stress
- Relative difficulty
  = absolute difficulty - power provided
- Perceived difficulty
  = absolute difficulty - (power provided + in-game experience)

(Adams, 2014)



Screenshot from Horizon Zero Dawn (Guerilla Games, 2017)

DDA - Game difficulty

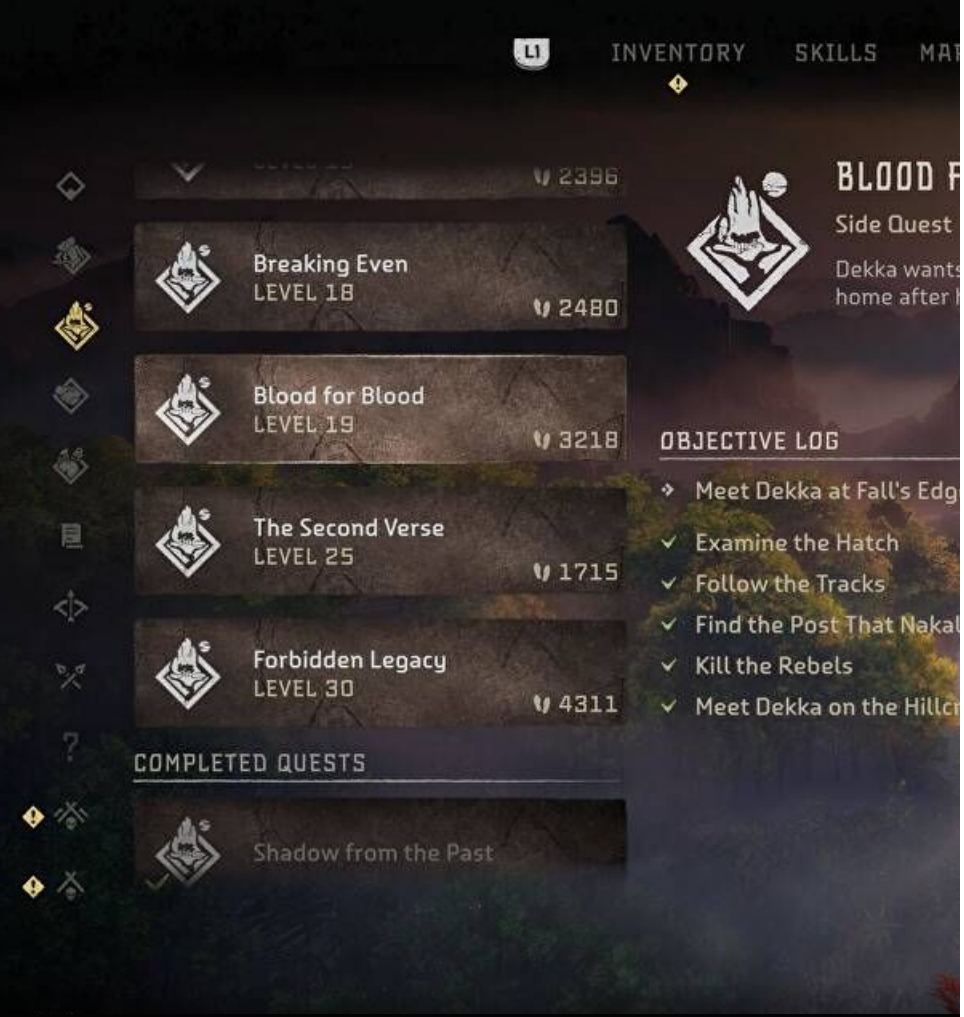
Difficulty types

- Absolute difficulty
  = Required intrinsic skill + Imposed stress
- Relative difficulty
  = absolute difficulty - power provided
- Perceived difficulty
  = absolute difficulty - (power provided + in-game experience)

In perspective to Game audio?

Reflect & Discuss with neighbor

(Adams, 2014)



Screenshot from Horizon Zero Dawn (Guerilla Games, 2017)

DDA - Game difficulty

Difficulty types

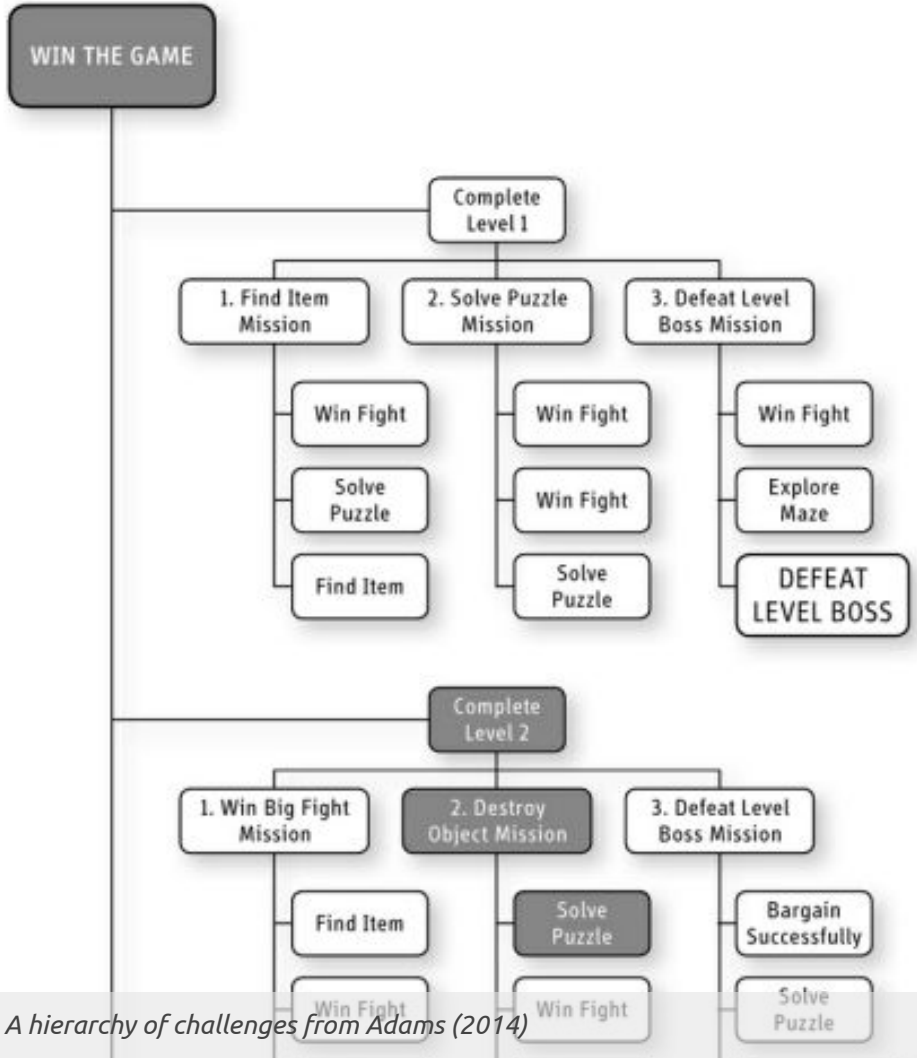
- Absolute difficulty
  = Required intrinsic skill + Imposed stress
- Relative difficulty
  = (power provided + in-game experience)
- Perceived difficulty
  = absolute difficulty - relative difficulty

Absolute- & relative difficulty → Game audio as (additional) **feedback channel**?

Perceived difficulty → Game audio - '**Affect**'

Imposed stress benadrukken met muziek.

(Adams, 2014, Huiberts & Tol, 2008)



A hierarchy of challenges from Adams (2014)

DDA - Game difficulty

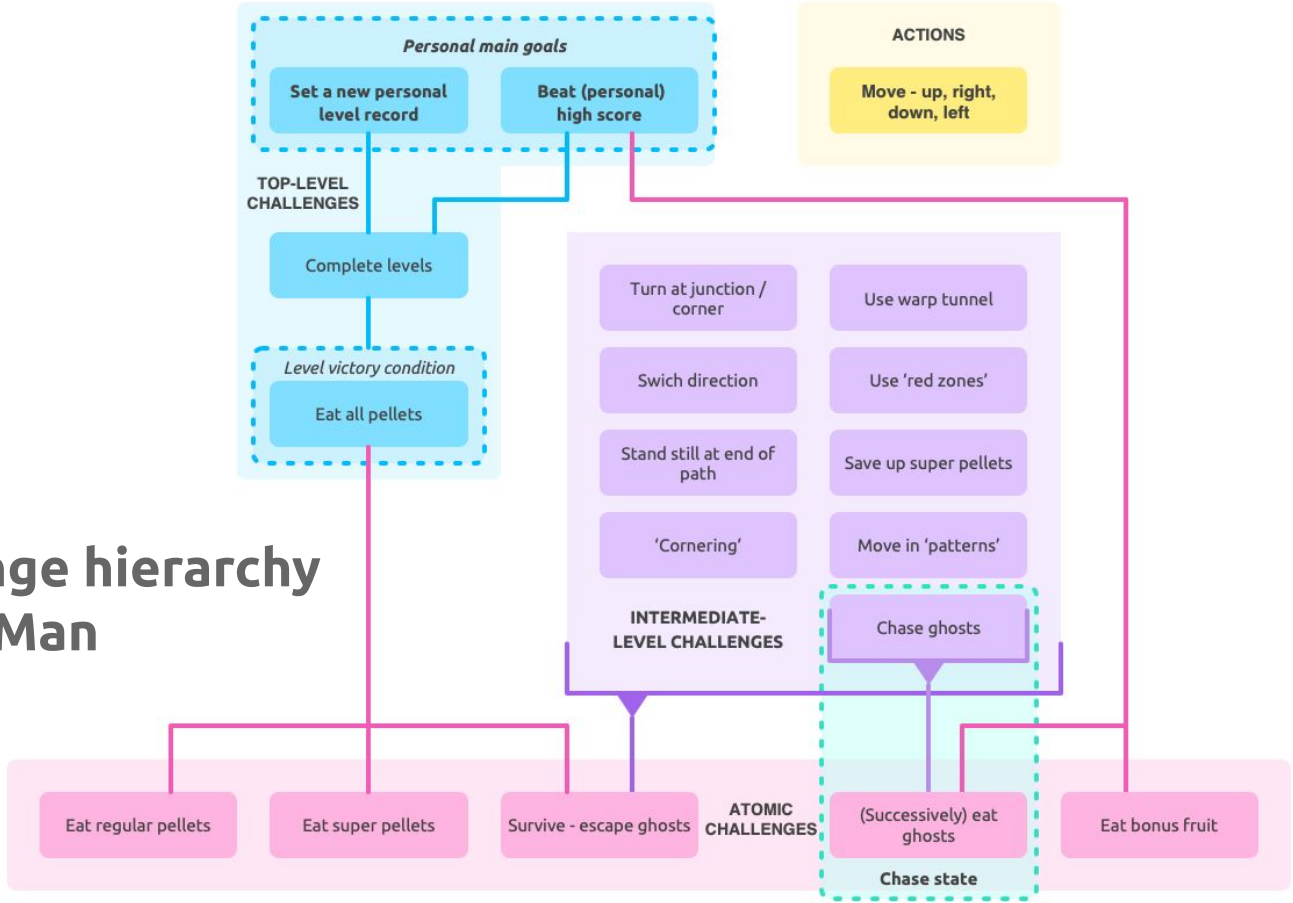
Challenge hierarchy

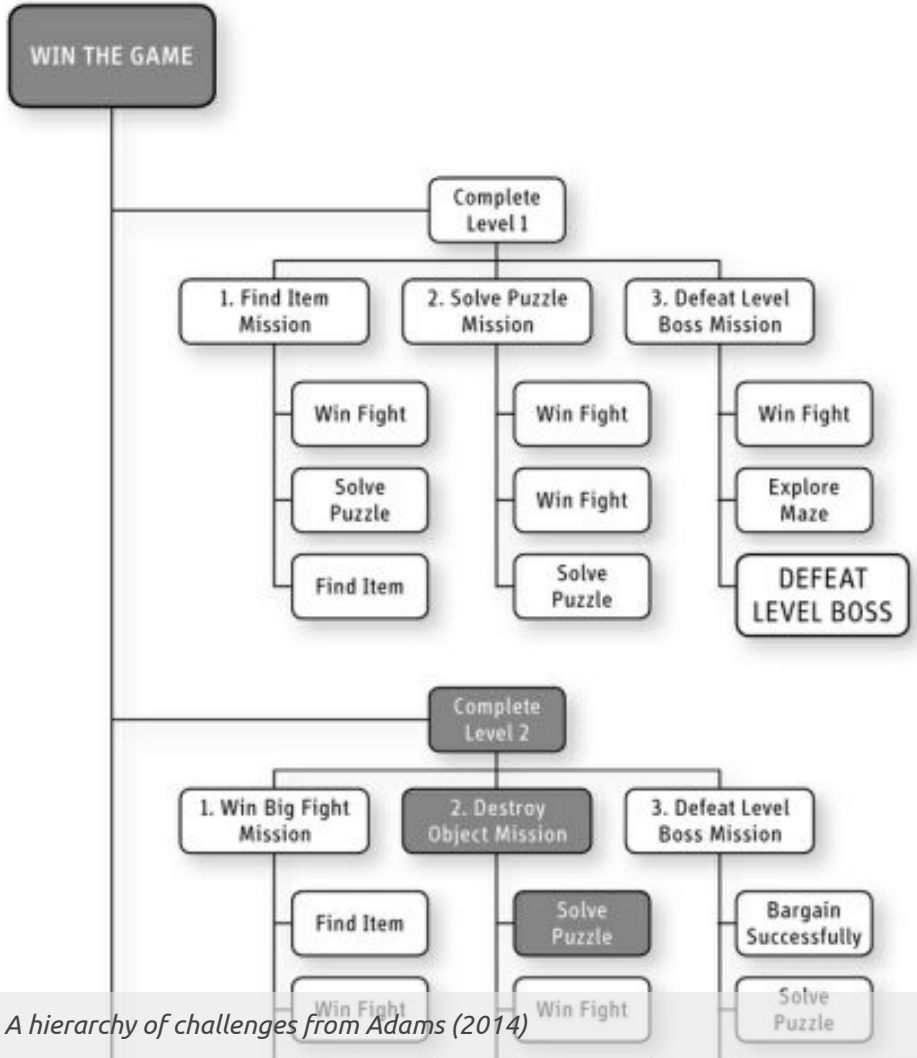
- Topmost-level
  - level / game victory condition
- Intermediate-level
  - implicit
  - strategy
- Lowest-level
  - atomic challenges

(Adams, 2014)

Game features selection

Challenge hierarchy of Pac-Man





A hierarchy of challenges from Adams (2014)

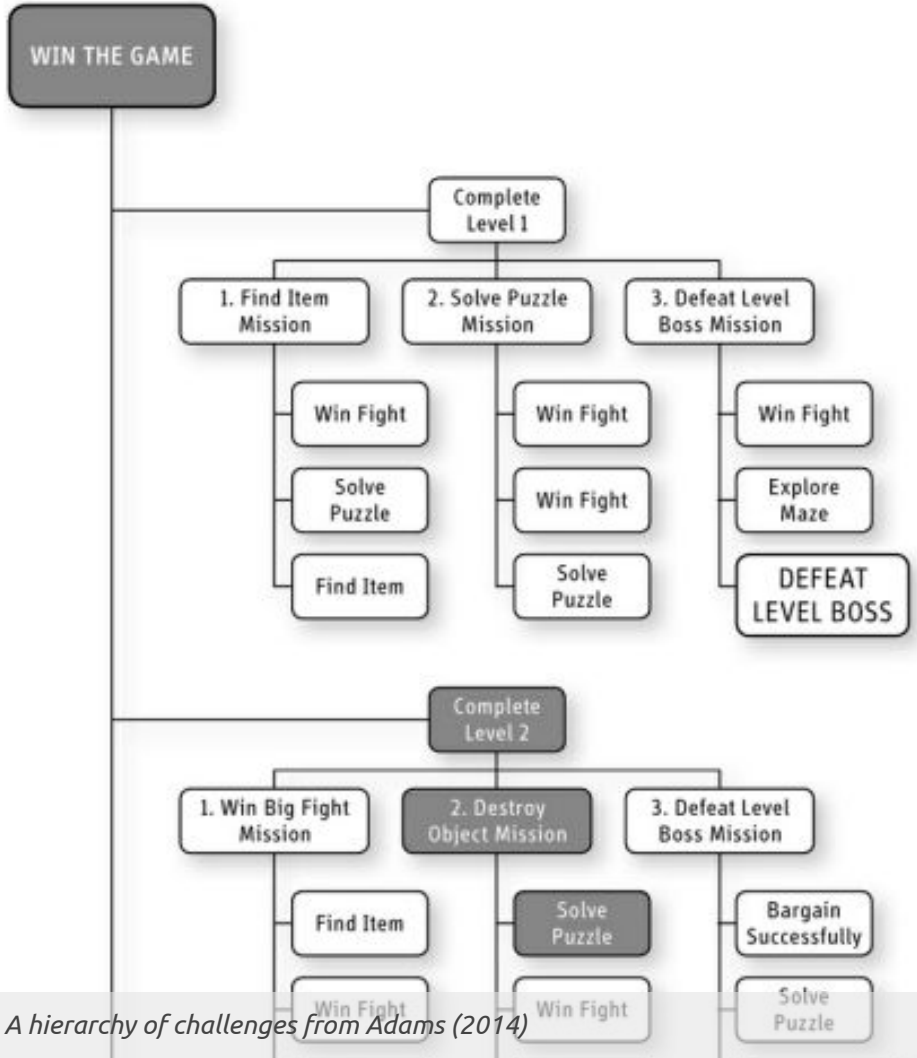
DDA - Game difficulty

Challenge hierarchy

- Topmost-level
  - level / game victory condition
- Intermediate-level
  - implicit
  - strategy
- Lowest-level
  - atomic challenges

In perspective to Game audio?

Reflect & Discuss with neighbor



A hierarchy of challenges from Adams (2014)

DDA - Game difficulty

Challenge hierarchy

- Topmost-level
  - level / game victory condition
- Intermediate-level
  - implicit
  - strategy
- Lowest-level
  - atomic challenges

Lowest-level - atomic challenges → sound design

Topmost-level → ‘Victory muziek’ bij behalen

Intermediate-level → reageren op de toegepaste strategieën

Noticeable rubber banding in racing games

Screenshot from Mario Kart 8 (Nintendo EAD, 2014)

DDA - Three basic requirements

1. **Quick identification** and adaptation to initial player's level
2. **Fast and close tracking** of and adaptation the player's improvement or falling level
3. **Unperceivable adaptation** process and game must remain believable

(Andrade et al., 2005, Adams & Dormans, 2012, Fernandes & Levieux, 2019)

Noticeable rubber banding in racing games

Screenshot from Mario Kart 8 (Nintendo EAD, 2014)

DDA - Three basic requirements

1. **Quick identification** and adaptation to initial player's level
2. **Fast and close tracking** of and adaptation the player's improvement or falling level
3. **Unperceivable adaptation** process and game must remain believable

In perspective to Game audio?

Reflect & Discuss with neighbor

Noticeable rubber banding in racing games

Screenshot from Mario Kart 8 (Nintendo EAD, 2014)

DDA - Three basic requirements

1. **Quick identification** and adaptation to initial player's level
2. **Fast and close tracking** of and adaptation the player's improvement or falling level
3. **Unperceivable adaptation** process and game must remain believable

Harde schakel van state → perceivable adaptation van de muziek, maar anders unperceivable?

Examples

Probabilistic method

“Dynamic difficulty adjustment through parameter manipulation for Space Shooter game”

1. Probabilistic methods

Space shooter game

- Assessment - **probabilistic calculation**
  - damage suffered in a given time (t)
  - amount of life

  → **estimated death at a future time**
- Adjustment - according to current **zone**
  - comfort zone
  - flow zone
  - discomfort zone
- Parameter manipulation
  - supply
  - demand

(Segundo et al., 2016)

“Dynamic difficulty adjustment through parameter manipulation for Space Shooter game”

1. Probabilistic methods

Space shooter game

- User study - three versions tested:
  - easy game
  - dynamic game
  - difficult game

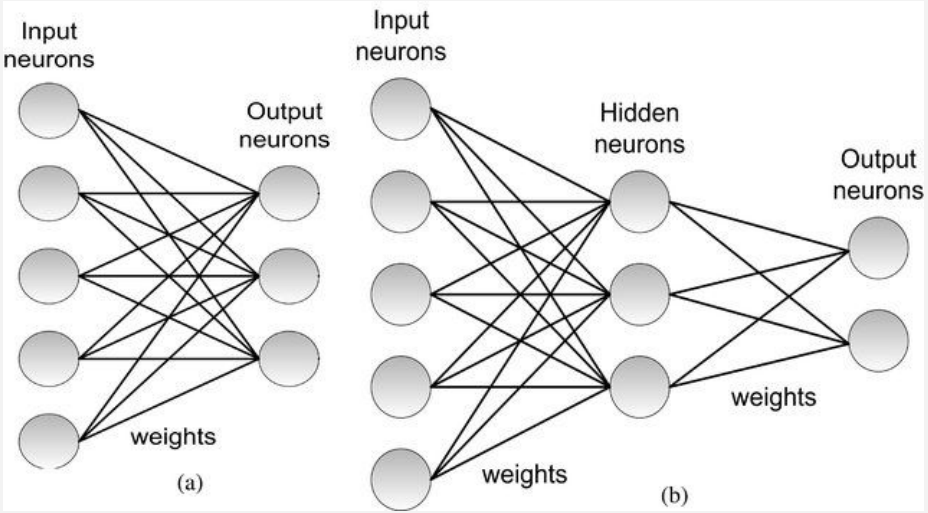
Results

- DDA achieved best results, regarding
  - playability
  - the will to play often

(Segundo et al., 2016)

Examples

Single & multilayered perceptrons



"(a) Architecture of a single layer perceptron. …

(b) Extension to a multi-layer perceptron including more than one layer of trainable weights. …

Each connection between two neurons is given by a certain weight."

(Camuñas-Mesa et al., 2019)

2. Single & multilayered perceptrons

Single perceptron vs multilayered perceptrons

- single perceptron = easier to analyze
- multilayered perceptron = higher accuracy of the function approximation

(Pedersen et al., 2009)

“Modeling Player Experience in Super Mario Bros”

2. Single & multilayered perceptrons

2D platform game

Single-neuron (perceptron) neural network:

- input:
  - gameplay (player behavior)
    e.g. # coins gathered, #deaths by jump in gap
  - design parameters (*controllable features*)
    - gap placement
    - gap size
    - gap spatial diversity
    - # direction switches
- output:
  - player’s emotions
    - frustration
    - fun
    - challenge

(Pedersen et al., 2009)

“Modeling Player Experience in Super Mario Bros”

2. Single & multilayered perceptrons

2D platform game

- Evolutionary preference learning
- 3 feature selection methods applied

  → *interested in the minimal feature subset that yields the highest performance*

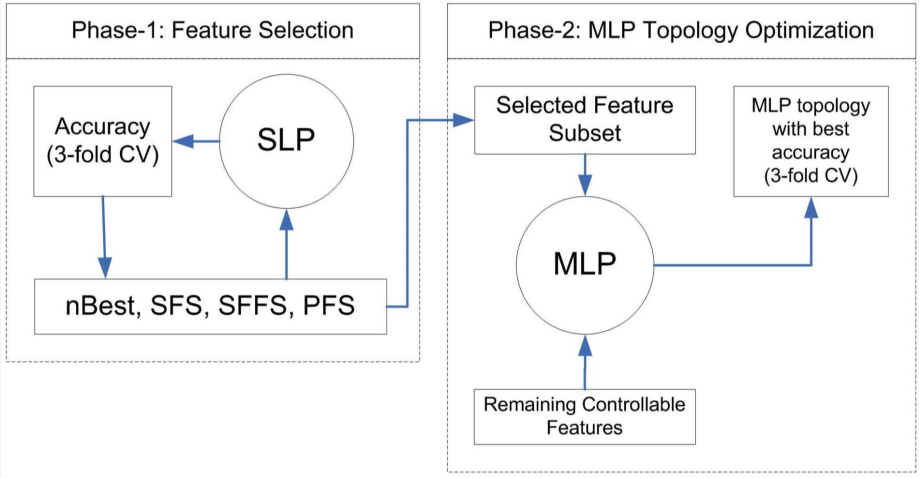
  → *subset used in follow up research*

Results:

- High accuracy of prediction of challenge (77.77%), frustration (88.66%), and fun (69.18%)

(Pedersen et al., 2009)

"Towards Automatic Personalized Content Generation for Platform Games"



2. Single & multilayered perceptrons

2D platform game

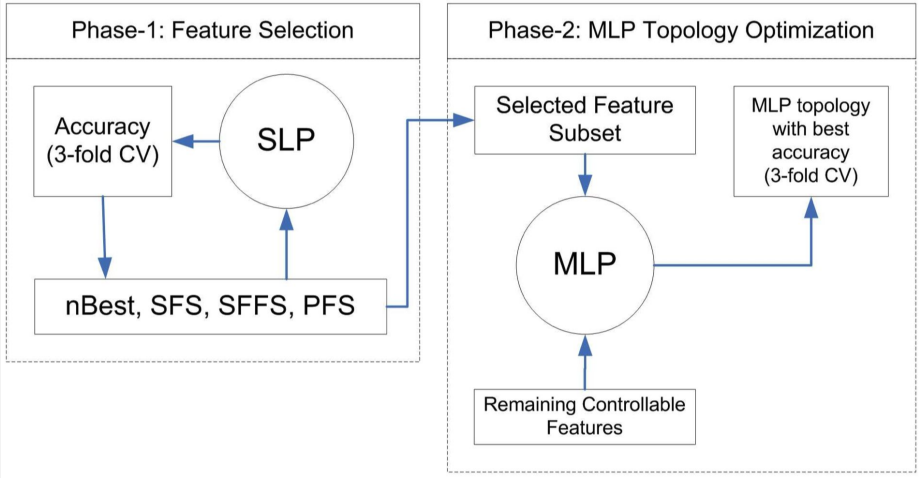
Follow up study (1)

Step 1. single layer perceptron:

- player experience **6 affective states** *(fun, challenging, boring, frustrating, predictable, anxious)*
- **4 feature selection methods** applied

(Pederson et al., 2010)

“Towards Automatic Personalized Content Generation for Platform Games”



2. Single & multilayered perceptrons

2D platform game

Step 2. **multi-layer perceptrons**

- **MLP topologies** of 2 hidden layers, up to 30 and 10 hidden neurons, *in total 330 different topologies*
- tested for each input vector
  - feature set selected from SLPs
  - remaining controllable features

(Pederson et al., 2010)

"Towards Automatic Personalized Content Generation for Platform Games"

The performance of the SLP networks is compared to MLPs built on selected features:

- *MLP_s - selected features*
- *MLP_c - controllable features*
- *MLP - selected and forced controllable features combined*

2. Single & multilayered perceptrons

2D platform game

Step 2. **multi-layer perceptrons**

- **MLP topologies** of 2 hidden layers, up to 30 and 10 hidden neurons, *in total 330 different topologies*
- tested for each input vector

	Fun	Challenge	Frustration	Predictability	Anxiety	Boredom
MLP Topology	6-2-2-1	9-3-1	8-10-10-1	10-4-6-1	8-5-3-1	6-21-10-1
SLP	69.18%	77.77%	88.66%	76.28%	70.63%	60.87%
MLP_s	74.84%	78.84%	87.33%	80.13%	75.40%	67.39%
MLP_c	66.04%	74.60%	78.67%	68.59%	73.02%	70.29%
MLP	74.21%	79.37%	91.33%	76.28%	77.78%	73.19%

(Pederson et al., 2010)

“Towards Automatic Personalized Content Generation for Platform Games”

2. Single & multilayered perceptrons

2D platform game

Follow up study (2)

- multilayered perceptron (MLP)
- **dynamic adaptation** of level generation parameters *(#gaps, avg. width gaps, gap placement, #direction switches)*
- small search space
  - 12000 configurations
  - find combination with max MLP output
  - allows real time level generation

(Shaker et al., 2010)

“Modeling Player Experience for Content Creation”

2. Single & multilayered perceptrons

2D platform game

Tested with:

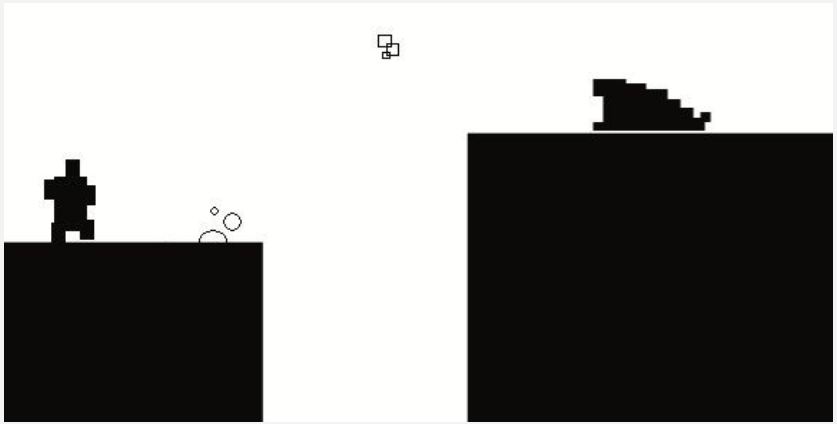
- 2 **AI agents** with different play styles
- 4 **participants** (pilot)

Results:

- “model robustly adapts to an individual player generalizing over various kinds of players” *(AI & participants)*

(Pederson et al., 2010)

“Polymorph: A Model for Dynamic Level Generation”



List of in-game actions

2. Single & multilayered perceptrons

2D platform game

Dynamic constructing levels with continually-appropriate challenge:

- multilayer perceptron model
- input:
  - gameplay features, *e.g. ‘whether the player died or completed a segment’*
  - level parameters:
    - occurrences of level components, *e.g. #upward-rising gaps*
    - occurrences of two-component adjacencies, *e.g. gap-enemy*

Results:

- accuracy of 66.4% for difficulty is less accurate 77% in Pederson et al. 2009
- however, examination of component combinations is possible

(Jennings-Teats et al., 2010)

“Polymorph: A Model for Dynamic Level Generation”

Feature	Correlation coefficient
Gap_Kill	0.7749
JumpUp	0.7095
Gap_Gap	0.6765
Gap_Avoid	0.6177
FlatGap	–0.5316
KillEnemy	–0.5222
Avoid	0.3818
JumpDownGap	0.3219
JumpUpGap	–0.0842
Avoid_Thwomp	0.0709

Ten most highly correlated features. Underscores indicate adjacent components.

2. Single & multilayered perceptrons

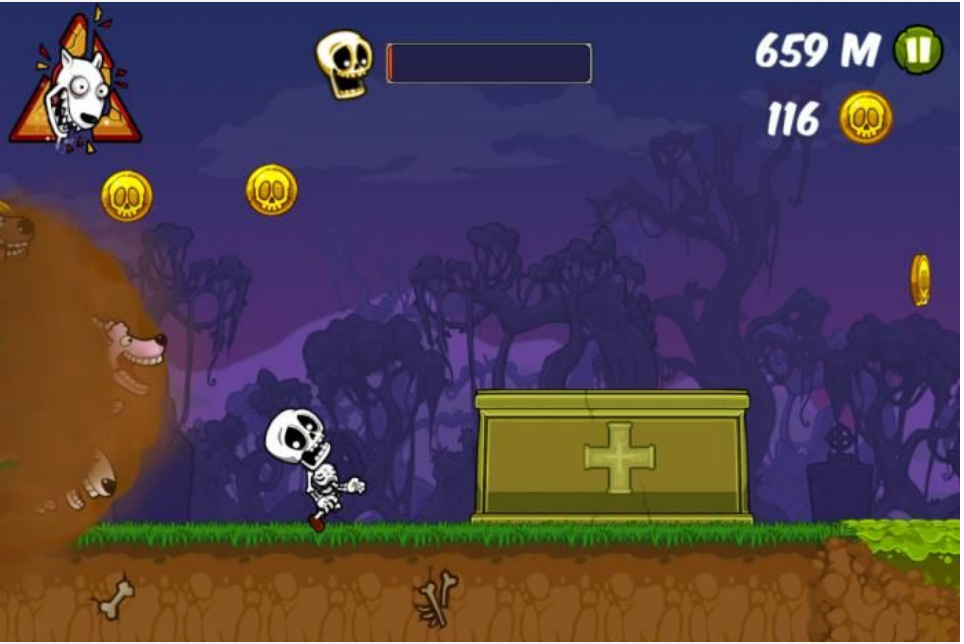
2D platform game

Results:

- Allows for examination of component combinations
- Accuracy of 66.4% for difficulty is less accurate 77% in Pederson et al. 2009

(Jennings-Teats et al., 2010)

“A Generic Framework for Procedural Generation of Gameplay Sessions”



A gameplay session of the game Boney the Runner.

2. Single & multilayered perceptrons

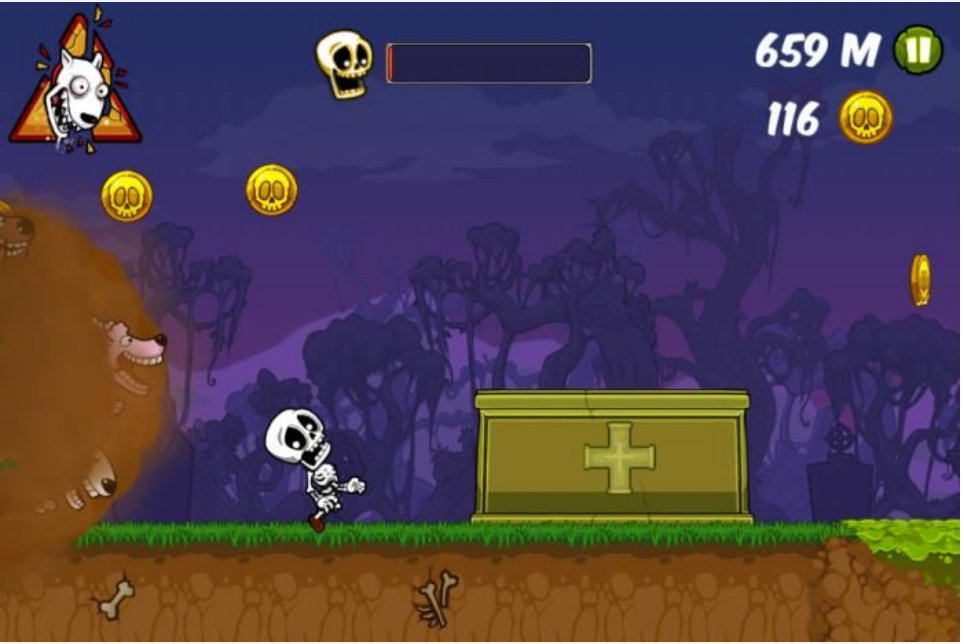
2D platform game

Content generation for endless games

- Chunk-based approach
  (chunk = playable segment with fixed width and heights)
- 4-step level generation process
  1. definition of difficulty curve
  2. chunk generation
  3. chunk evaluation
  4. chunk insertion

(Carvalho et al., 2013)

“A Generic Framework for Procedural Generation of Gameplay Sessions”



A gameplay session of the game Boney the Runner.

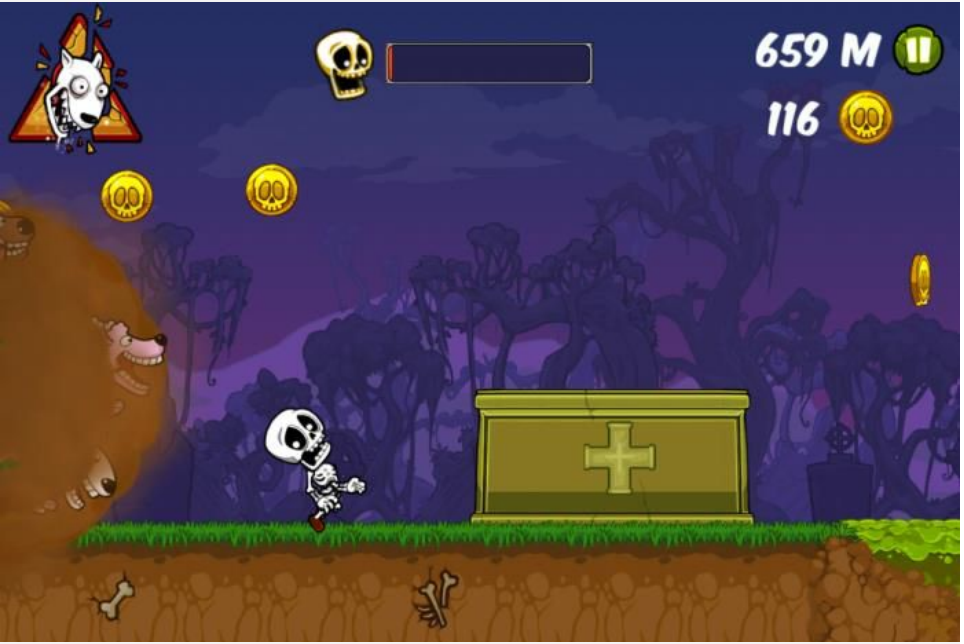
2. Single & multilayered perceptrons

2D platform game

- Difficulty based on time that player has lost on each chunk
- **Two multilayered networks** *(1 hidden layer)*:
  - A. input: controllable features
  - B. input: controllable features & non-controllable features.
  - A & B output: chunk difficulty
- **Network A: initial stages of development**, only access to controllable features from the chunks

(Carvalho et al., 2013)

“A Generic Framework for Procedural Generation of Gameplay Sessions”



A gameplay session of the game Boney the Runner.

2. Single & multilayered perceptrons

2D platform game

Results:

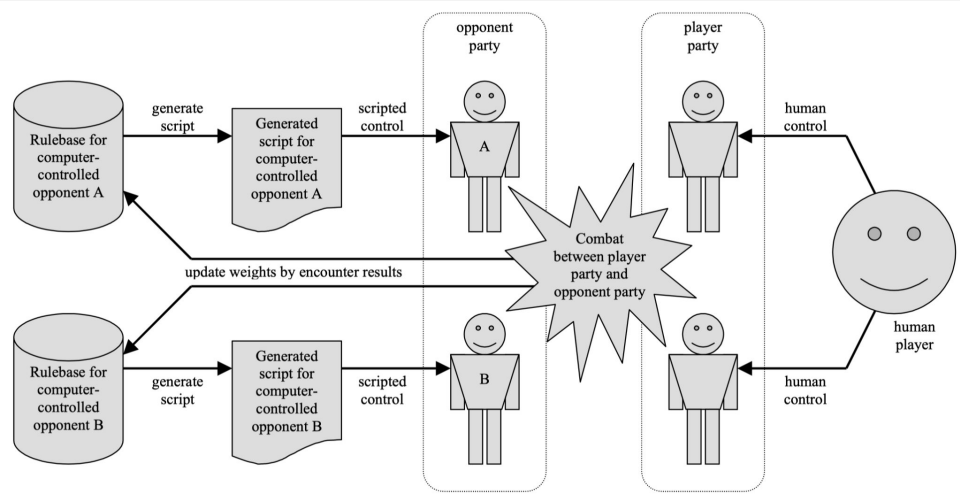
- 70% accuracy network A. *(controllable features only)*
- 90% accuracy network B
- 52% accuracy human designer

(Carvalho et al., 2013)

Examples

Dynamic scripting

“Online Adaptation Of Game Opponent Ai With Dynamic Scripting”



The dynamic scripting process.

3. Dynamic scripting

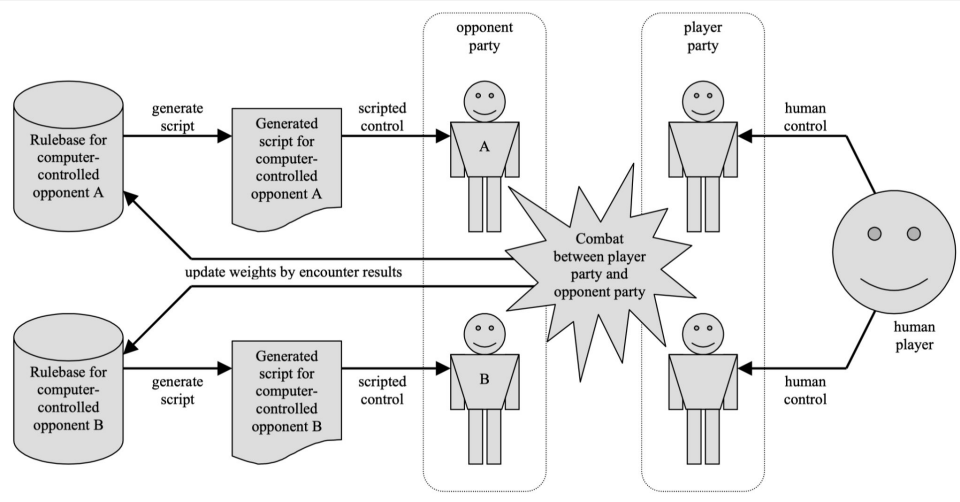
‘Complex’ RPG

Rulebases *(1 for each opponent type)* used to dynamically generate **scripts for NPCs**

- after an encounter → **weights are changed**
  - weights of rules that lead to success are increased
  - weights of rules that lead to failure are decreased
  - weights of other rules → updated *maintain unchanged sum of weights*
- Weight change size determined by **weight-update function**

(Spronck et al., 2004a)

“Difficulty Scaling of Game AI”



The dynamic scripting process.

3. Dynamic scripting

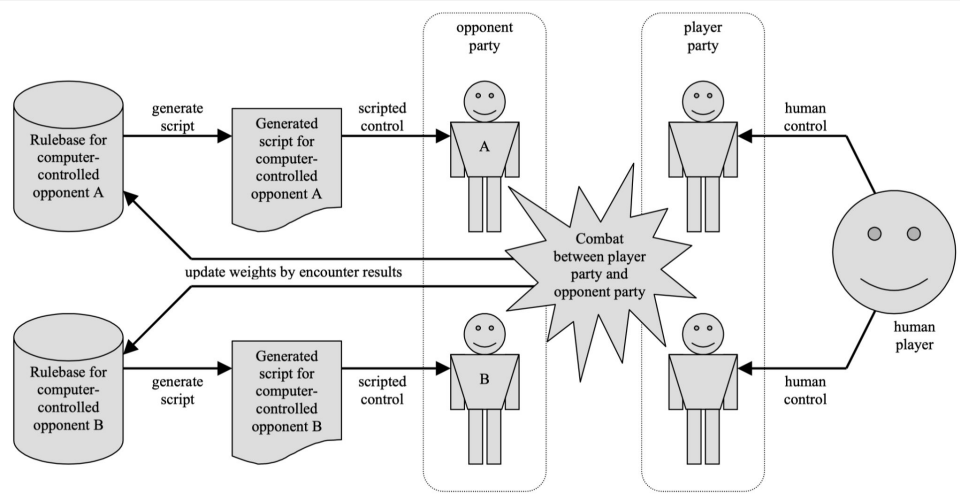
‘Complex’ RPG

To scale difficulty level of the game AI to that of the player - three enhancements:

- High-fitness penalizing
- Weight clipping
- Top culling

(Spronck et al., 2004b)

“Difficulty Scaling of Game AI”



The dynamic scripting process.

3. Dynamic scripting

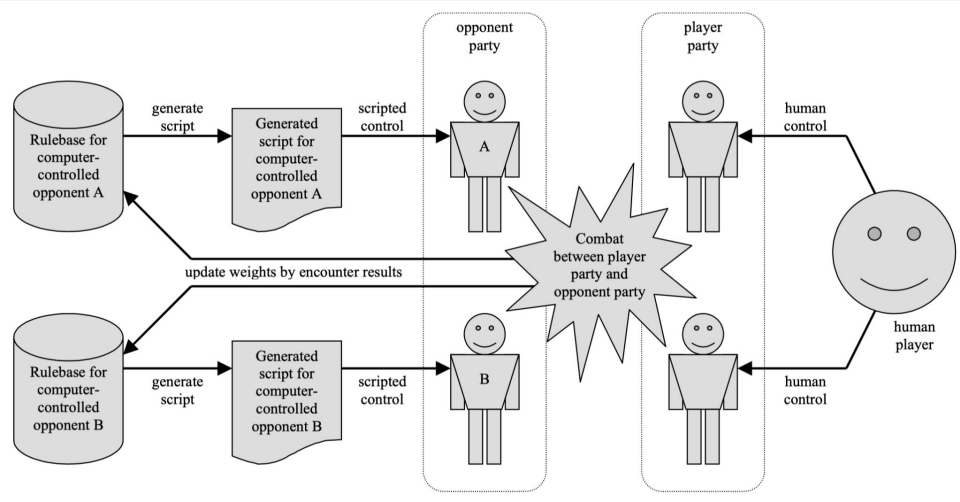
‘Complex’ RPG

To scale difficulty level of the game AI to that of the player - three enhancements:

- High-fitness penalizing
  - highest rewards to mediocre fitness values
  - lower rewards or penalties to high fitness values
- Weight clipping
- Top culling

(Spronck et al., 2004b)

“Difficulty Scaling of Game AI”



The dynamic scripting process.

3. Dynamic scripting

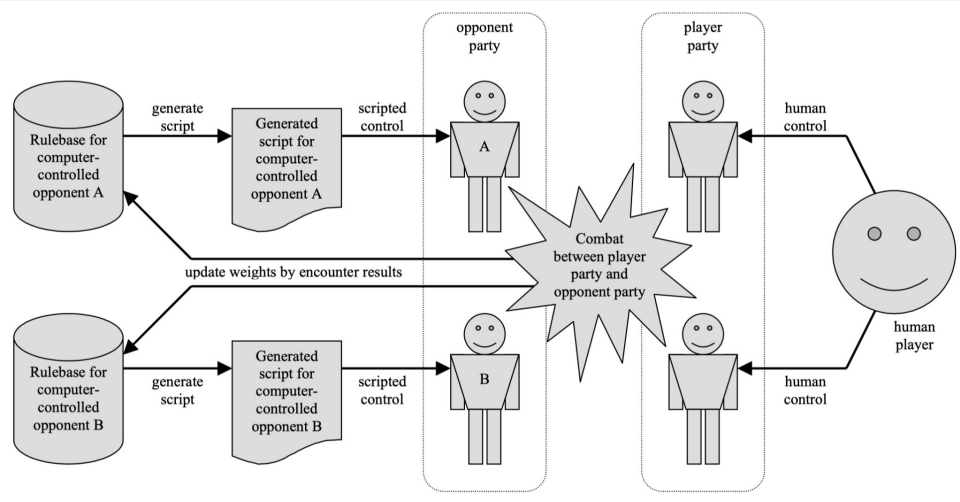
‘Complex’ RPG

To scale difficulty level of the game AI to that of the player - three enhancements:

- High-fitness penalizing
- Weight clipping
  - automatically varying the maximum weight value
  - high max weight values → lead to scripts that are close to optimal
  - low max weight values → enforces high diversity in generated scripts, most not optimal
- Top culling

(Spronck et al., 2004b)

“Difficulty Scaling of Game AI”



The dynamic scripting process.

3. Dynamic scripting

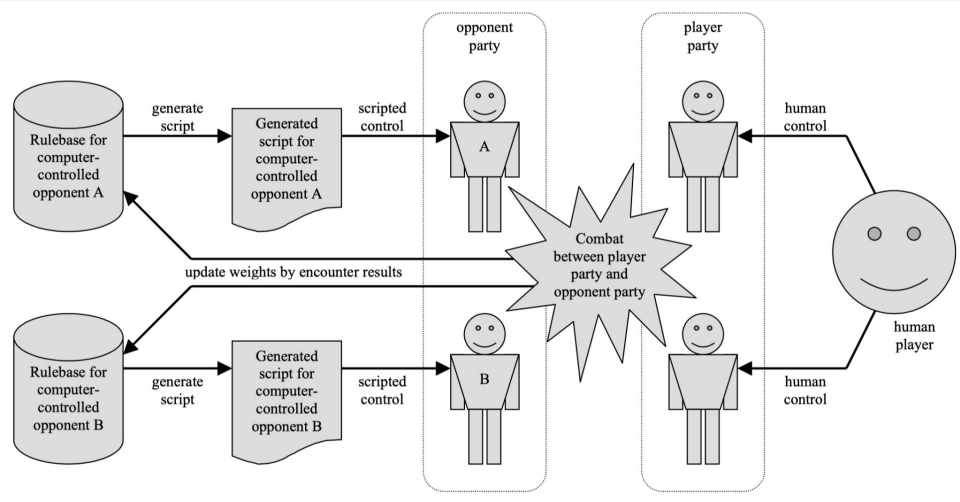
‘Complex’ RPG

To scale difficulty level of the game AI to that of the player - three enhancements:

- High-fitness penalizing
- Weight clipping
- Top culling:
  - similar to weight clipping
  - allows weights to grow beyond the max value → rules will not be selected anymore *(thus weaker tactics will be selected)*

(Spronck et al., 2004b)

“Difficulty Scaling of Game AI”



The dynamic scripting process.

3. Dynamic scripting

‘Complex’ RPG

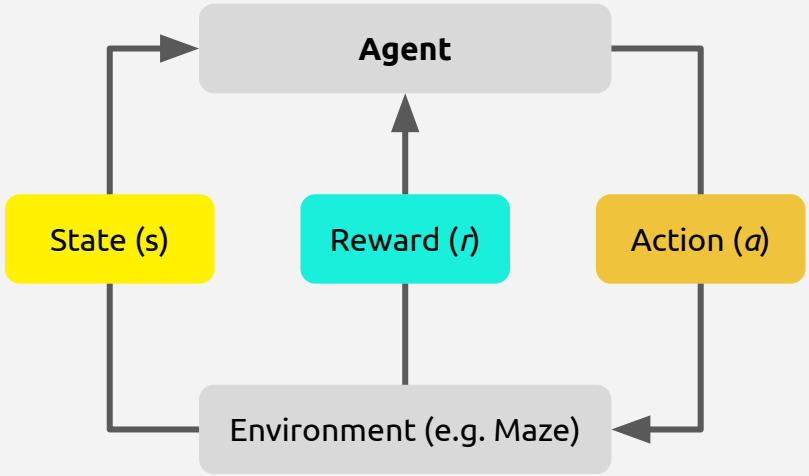
Results:

- DDA by dynamic scripting is effective
- Weight clipping & Top culling successful
- high-fitness penalizing unsuccessful

(Spronck et al., 2004b)

Examples

Reinforcement Learning



A reinforcement learning example (based on image from Yannakakis & Togelius, 2018)

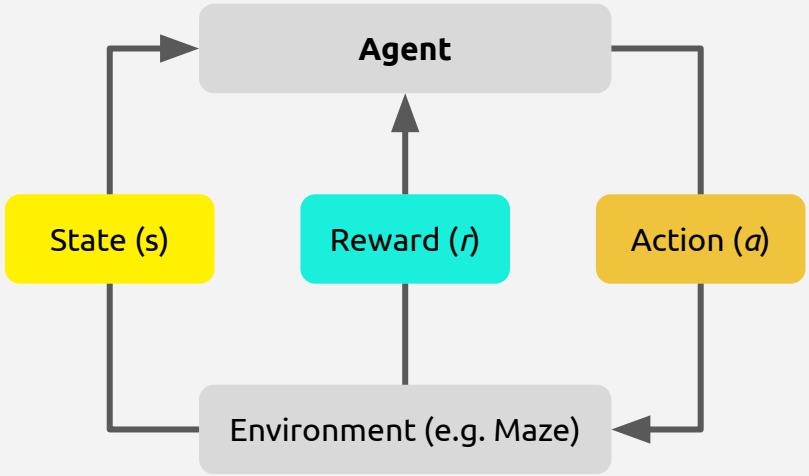
5. Reinforcement Learning

Theoretical background

*"Reinforcement learning … refers to methods that solve reinforcement learning problems, where a **sequence of actions is associated with positive or negative rewards** …"*

*"... inspired by … the way humans and animals learn to take decisions via (positive or negative) **rewards received by their environment**."*

(Yannakakis & Togelius 2018)



A reinforcement learning example (based on image from Yannakakis & Togelius, 2018)

5. Reinforcement Learning

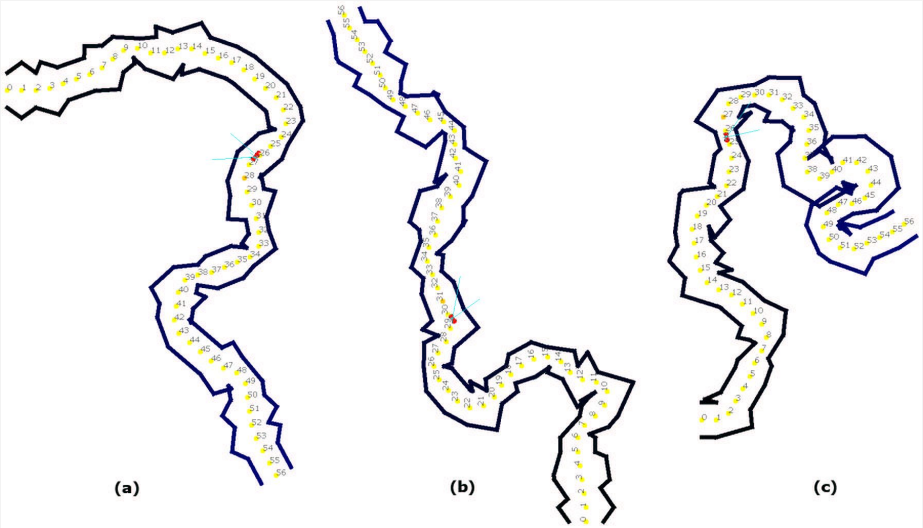
Theoretical background

*"Through the **continuous interaction** between the agent and its environment, the agent gradually learns to select actions that **maximize its sum of rewards**"*

At point in time *t*

- Agent:
  - State *s*
  - Decides on action *a*
    from all available actions in current state
- Environment delivers response:
  - Immediate reward *r*

“Making Racing Fun Through Player Modeling and Track Evolution”



5. Reinforcement Learning

Racing game

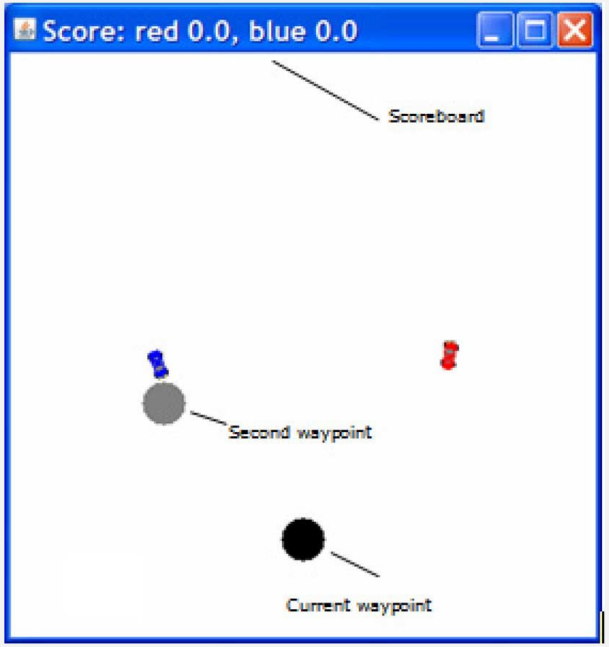
Adaptive game AI can create varied gaming experiences for different playing styles:

+ adds interest
+ adds repeatability of play

E.g. evolutionary algorithms to **create racing tracks tailor-mad**e to maximize enjoyment of individual player.

(Togelius et al., 2006)

“Dynamic Game Difficulty Scaling Using Adaptive Behavior-Based AI”



Playing area for simulated car racing.

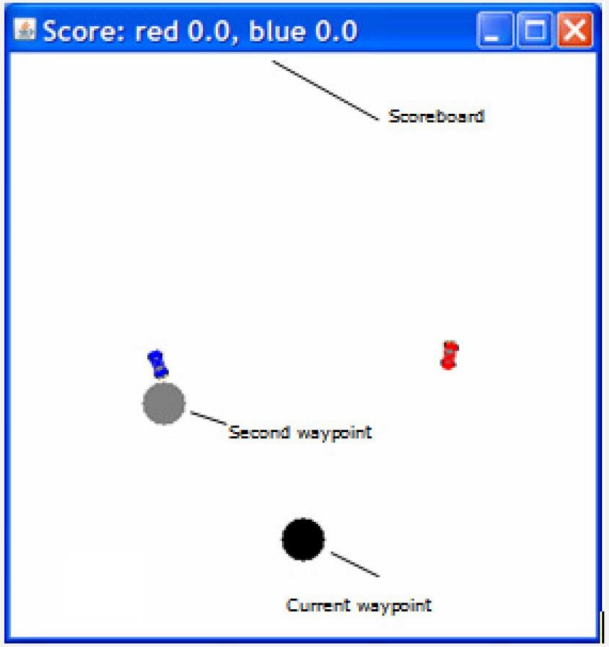
5. Reinforcement Learning

Racing game

- Objective = development of an adaptive game AI:
  - promotes **even game** *(more enjoyable)* rather than beating opponents.
  - **adapts its behavior** according to opponent’s moves
  - **real time** difficulty scaling

(Hagelbäck & Johansson, 2009, Tan et al., 2011)

“Dynamic Game Difficulty Scaling Using Adaptive Behavior-Based AI”



Playing area for simulated car racing.

5. Reinforcement Learning

Racing game

Two adaptive algorithms presented, use ideas from:

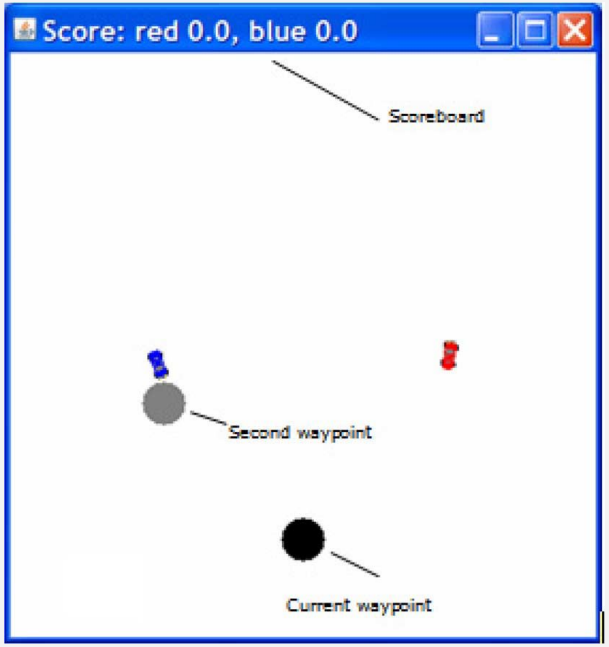
- reinforcement learning
- evolutionary computation

Two indicators as **measure of entertainment value**:

- winning percentage difference
  (|W-L| and D to be minimized, where W, L, and D are wins, losses, and draws)
- mean score difference
  (|s1-s2| to be minimized and max(s1,s2), where s are scores of players 1 and 2)

(Tan et al., 2011)

“Dynamic Game Difficulty Scaling Using Adaptive Behavior-Based AI”



Playing area for simulated car racing.

5. Reinforcement Learning

Racing game

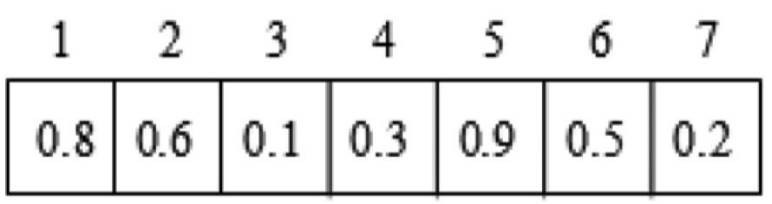
Artificial Stupidity: fine tuning of a game AI such that it provides the player with an entertaining experience **by deliberately making mistakes** *(intentional, but plausible)*

Controller adaptation approach:

- Start with **‘overdesigned’ game AI** :
  - good set of game behavior components
  - able to defeat player
- **Estimate player’s competency**
- Progressively **select subset** of behavior components

(Liden, 2004, Tan et al., 2011)

“Dynamic Game Difficulty Scaling Using Adaptive Behavior-Based AI”



Representation of the chromosome used in the Adaptive Uni-Chromosome Controller

5. Reinforcement Learning

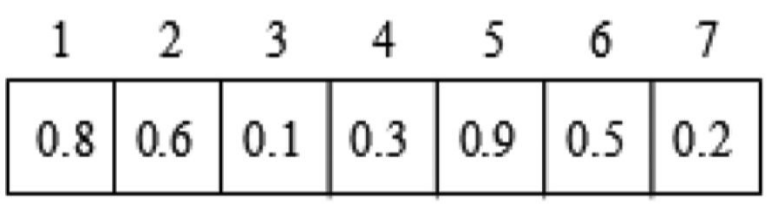
Racing game

Adaptive Uni-Chromosome Controller (AUC)

- online training & adaptation process
- chromosome:
  - encodes the **probability** ([0, 1]) **for activating a behavior component** after waypoint passing
- assumption complement of winning strategy == losing strategy
- ruleset updates:
  - after controller loss / draw → winning chromosome
  - after controller winning → complement of winning chromosome

(Tan et al., 2011)

“Dynamic Game Difficulty Scaling Using Adaptive Behavior-Based AI”



1	2	3	4	5	6	7
0.8	0.6	0.1	0.3	0.9	0.5	0.2

Representation of the chromosome used in the Adaptive Uni-Chromosome Controller.

5. Reinforcement Learning

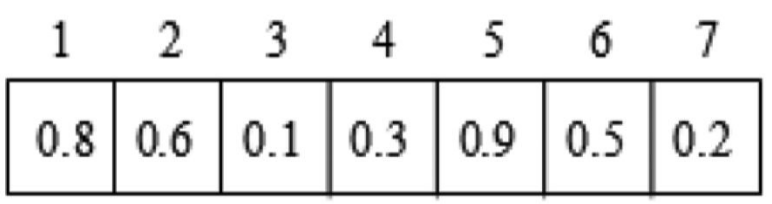
Racing game

Adaptive Duo-Chromosome Controller (ADC)

- maintains two sets of chromosome:
  - winning chromosome
  - losing chromosome
- assumption complement of winning strategy != losing strategy

(Tan et al., 2011)

“Dynamic Game Difficulty Scaling Using Adaptive Behavior-Based AI”



1	2	3	4	5	6	7
0.8	0.6	0.1	0.3	0.9	0.5	0.2

Representation of the chromosome used in the Adaptive Uni-Chromosome Controller.

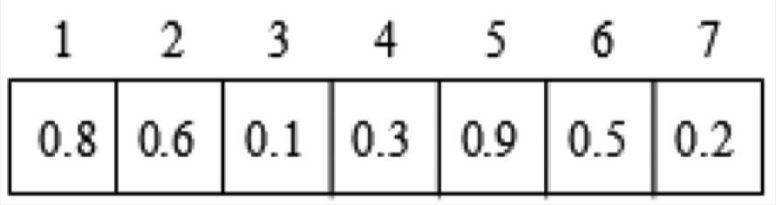
5. Reinforcement Learning

Racing game

AUC and ADC tested against static controllers to simulate players with different styles of play.

(Tan et al., 2011)

“Dynamic Game Difficulty Scaling Using Adaptive Behavior-Based AI”



1	2	3	4	5	6	7
0.8	0.6	0.1	0.3	0.9	0.5	0.2

Representation of the chromosome used in the Adaptive Uni-Chromosome Controller.

5. Reinforcement Learning

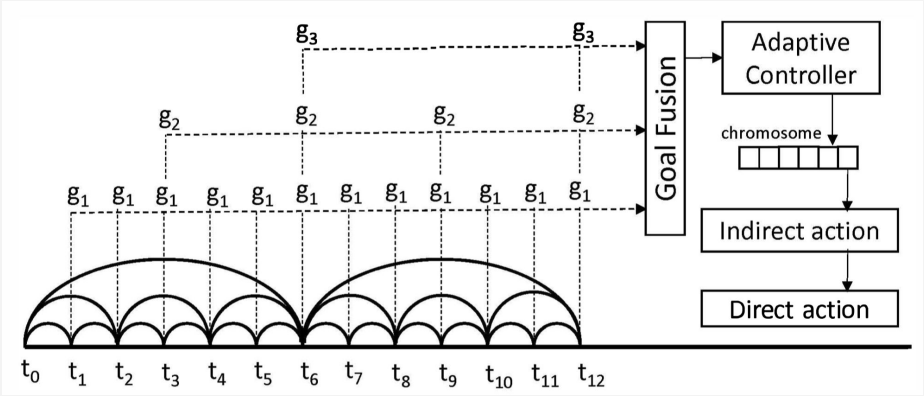
Racing game

Results

- able to learn proper set of behavior components
- able to generalize satisfactory for various opponents
- achieved score differences of 4 or less in > 72% of the games
- scattered wins and losses
- final set of chromosomes showed various component combinations to deal with different styles
- ADC - lower #drawn games

(Tan et al., 2011)

“MPRL: Multiple-Periodic Reinforcement Learning for Difficulty Adjustment in Rehabilitation Games”



The architecture of MPRL where different goals are evaluated in separate periodS.

5. Reinforcement Learning

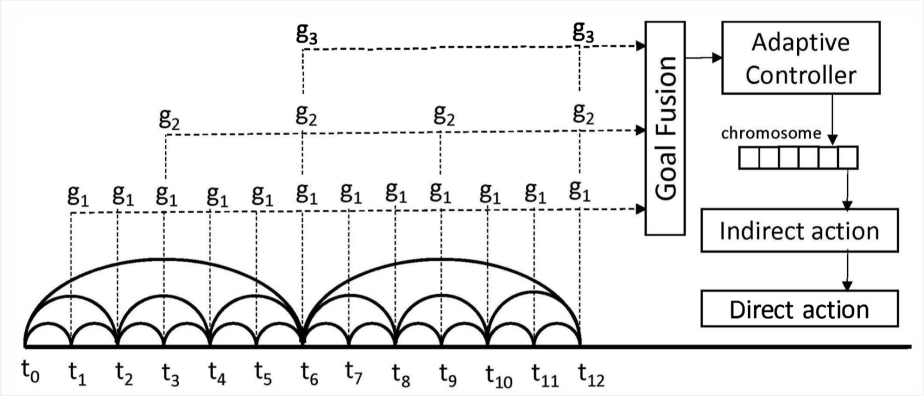
Rehabilitation game

Personalized DDA method for rehab based on real-time patient’s skills

- DDa was argued to be a multiple-goal problem
- multiple objectives:
  - satisfaction
  - engagement
  - perceiving a progress
- Multiple-**Objective** Reinforcement learning insufficient due to different measurement periods of objectives
  → proposed Multiple-**Periodic** Reinforcement Learning

(Sekhavat, 2017),)

“MPRL: Multiple-Periodic Reinforcement Learning for Difficulty Adjustment in Rehabilitation Games”



The architecture of MPRL where different goals are evaluated in separate periodS.

5. Reinforcement Learning

Rehabilitation game

Results

- Multiple-**Periodic** Reinforcement Learning performed better than Multiple-**Objective** Reinforcement learning methods in user satisfaction and enhancing patient motor skills

(Sekhavat, 2017),)

6.

6. Upper Confidence Bound for Trees and Artificial Neural Networks

6. Upper Confidence Bound for Trees and Artificial Neural Networks

Theoretical background

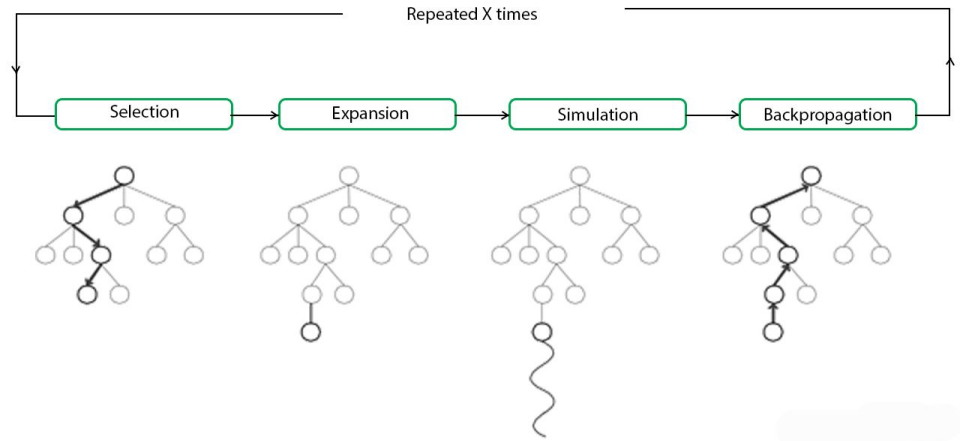


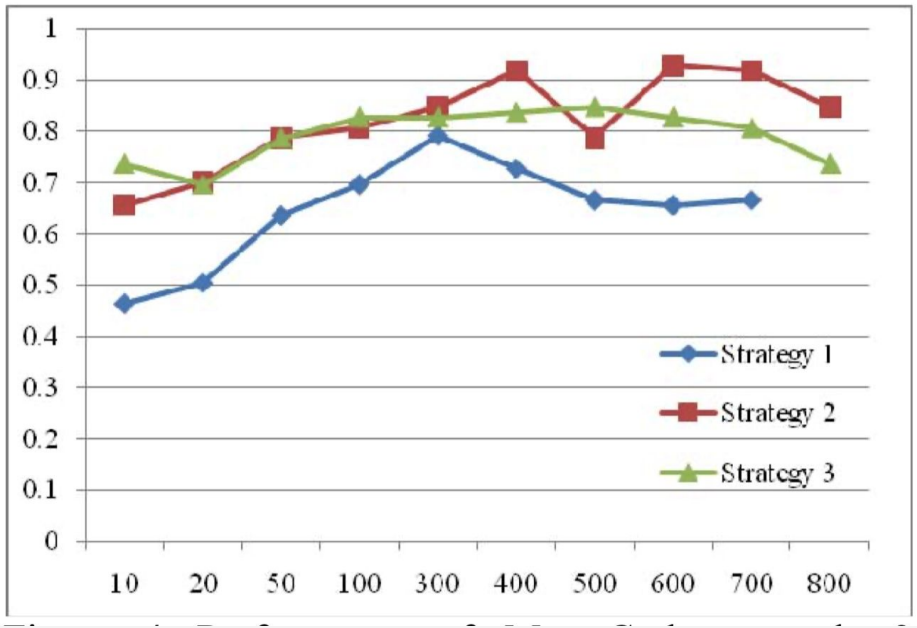
image from Duarte et al.,2020

Monte Carlo tree search:

- a probabilistic and heuristic driven search algorithm
- combination of
  - classic tree search implementations
  - machine learning principles of reinforcement learning

(Kocsis & Szepesviri, 2006)

“To Create Intelligent Adaptive Game Opponent by Using Monte-Carlo for Tree Search”



Simplified Interface of Video Game Pac-Man

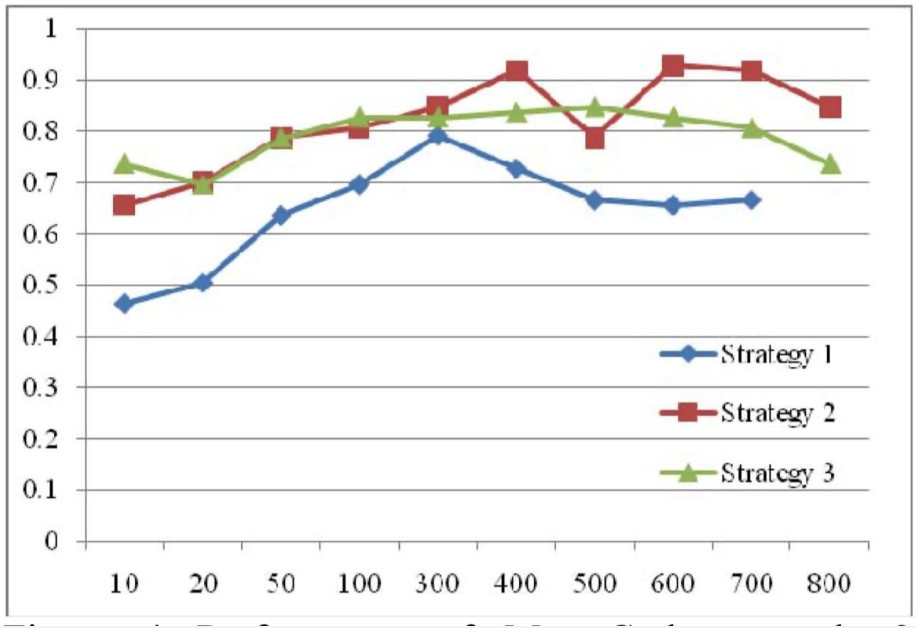
6. Upper Confidence Bound for Trees and Artificial Neural Networks

Predator-prey game

- MCTS **performance** significantly **correlates** with the **duration of simulation**
  - after time threshold value - performance decreases down to a **stable state**

(Yang et al., 2009)

“To Create Intelligent Adaptive Game Opponent by Using Monte-Carlo for Tree Search”



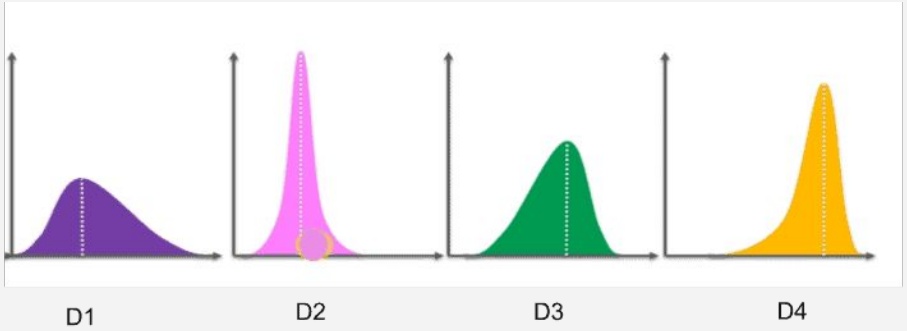
Simplified Interface of Video Game Pac-Man

6. Upper Confidence Bound for Trees and Artificial Neural Networks

Predator-prey game

- MCTS performance significantly correlates with the duration of simulation
  - after time threshold value - performance decreases down to a stable state
- Not able to use MCTS realtime due to computational intensiveness
  - Transform MCTS → ANN
    Three ANNs were trained with MCTS created data against three different types of simulated players

(Yang et al., 2009)



Four different distributions to demonstrate the Multi-Armed Bandit Problem

image from https://www.mltut.com/upper-confidence-bound-reinforcement-learning-super-easy-guide

6. Upper Confidence Bound for Trees and Artificial Neural Networks

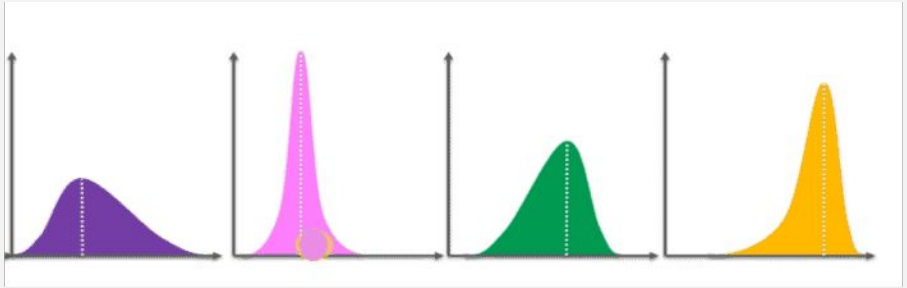
Theoretical background

Upper confidence Bound solves the **Multi-Armed Bandit Problem**

- multiple **unknown distributions**
- aim: find the best distribution

e.g. multiple slot machines and find the machine with highest winnings - how much time should you:

- **explore**, which machine is the best?
- **exploit**, the machine you think is best?



Four different distributions to demonstrate the Multi-Armed Bandit Problem

image from https://www.mltut.com/upper-confidence-bound-reinforcement-learning-super-easy-guide

6. Upper Confidence Bound for Trees and Artificial Neural Networks

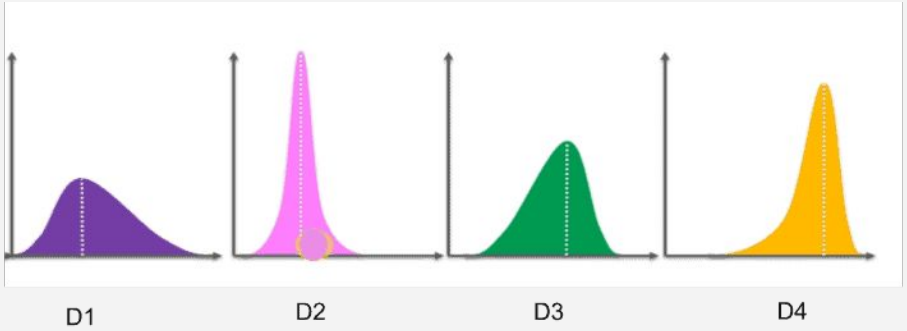
Theoretical background

Epsilon Greedy strategy

e.g. with Epsilon 10%:

- 10% chance to explore
- 90% chance to exploit current learned distributions *(choose 'the best' distribution up to now)*

$$\pi(a|s) = \begin{cases} \text{with probability } \epsilon, & \text{Do } \textbf{Exploration}\text{: } \textbf{\textit{Randomly}} \text{ pick an action for this state} \\ \text{with probability } 1-\epsilon, & \text{Do } \textbf{Exploitation}\text{: } \textbf{\textit{Greedily}} \text{ pick an action for this state} \\ & a = \arg\max_a Q(s,a) \end{cases}$$



Four different distributions to demonstrate the Multi-Armed Bandit Problem

image from https://www.mltut.com/upper-confidence-bound-reinforcement-learning-super-easy-guide

6. Upper Confidence Bound for Trees and Artificial Neural Networks

Theoretical background

Upper confidence Bound

Epsilon Greedy Strategy exploitation is **used 'extended' with upper confidence bound**, to **balance exploration and exploitation**:

low exploration for action a?
→ denominator is lower
→ quantity second term is higher
→ higher chance to explore a

$$A_t = \arg\max_a \left[Q_t(a) + c\sqrt{\frac{\log t}{N_t(a)}} \right]$$

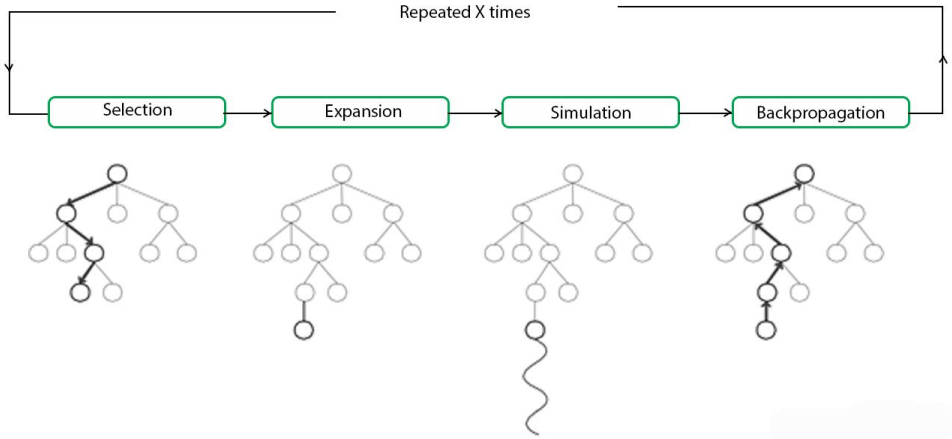


image from Duarte et al.,2020

6. Upper Confidence Bound for Trees and Artificial Neural Networks

Theoretical background

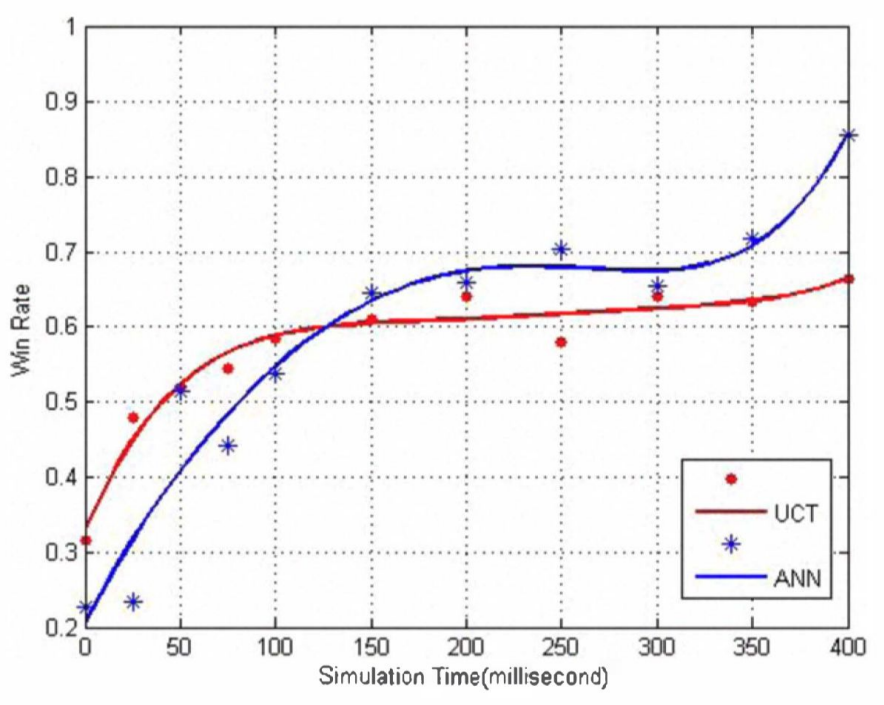
Monte Carlo tree search:

- a probabilistic and heuristic driven search algorithm
- combination of
  - classic tree search implementations
  - machine learning principles of reinforcement learning

Upper confidence Bound for Trees (UCT) one of the most popular and generally effective **Monte Carlo tree search** (MCTS).

(Kocsis & Szepesviri, 2006)

"To create DDA by the approach of ANN from UCT-created data"



DDA from UCT and DDA from ANN

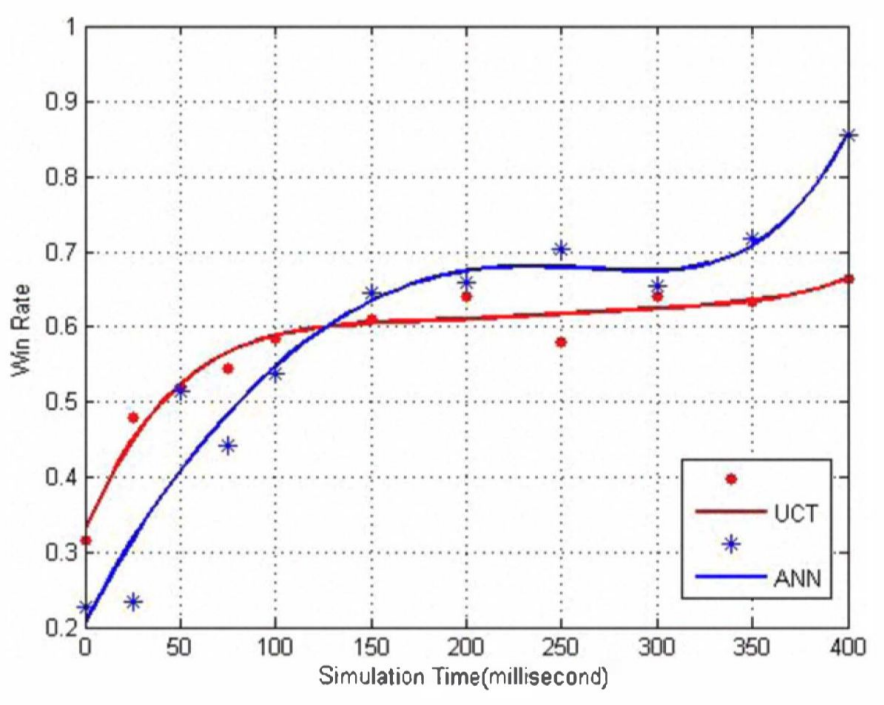
6. Upper Confidence Bound for Trees and Artificial Neural Networks

Predator-prey game

- UCT **performance** significantly correlates with the **duration of simulation**
  - usable for 'DDA agent' by limiting computation time!
- Not able to use UCT realtime due to computational intensiveness
  - Transform UCT → ANN
    ANNs were trained with UCT created data with different simulation time

(Li et al., 2010)

"To create DDA by the approach of ANN from UCT-created data"



DDA from UCT and DDA from ANN

6. Upper Confidence Bound for Trees and Artificial Neural Networks

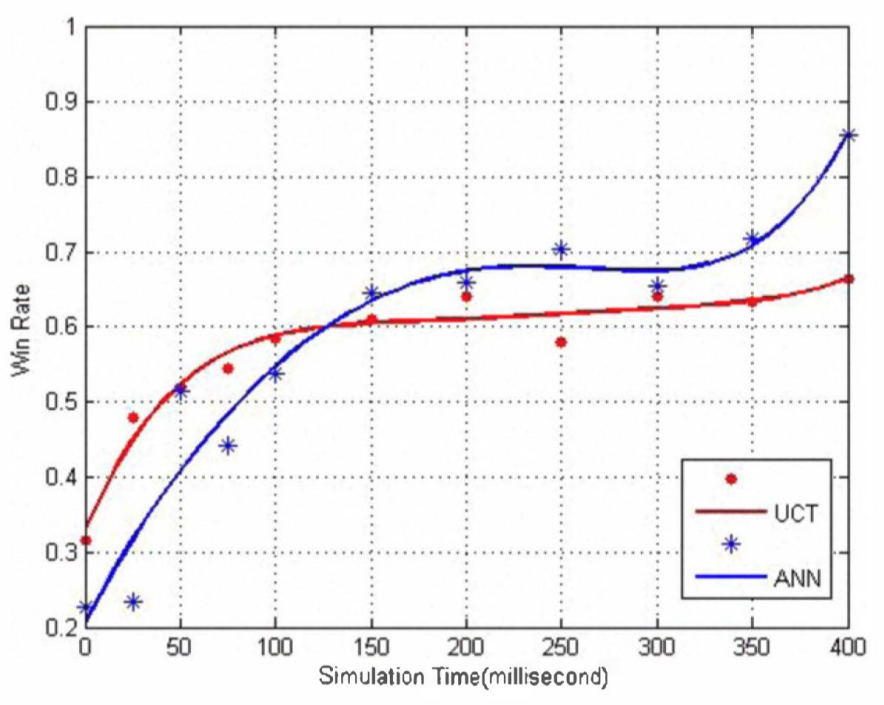
Predator-prey game

- ANN performance depends on training data quality
  - insufficient incidences for certain routes
    → ANN training remains poor
  - ample incidences for all routes
    → trained ANN performs well

- **Increasing** the growth of s**imulation time**
  → the UCT data achieve greater precision
  → **better trained ANN**

(Li et al., 2010)

"To create DDA by the approach of ANN from UCT-created data"



DDA from UCT and DDA from ANN

6. Upper Confidence Bound for Trees and Artificial Neural Networks

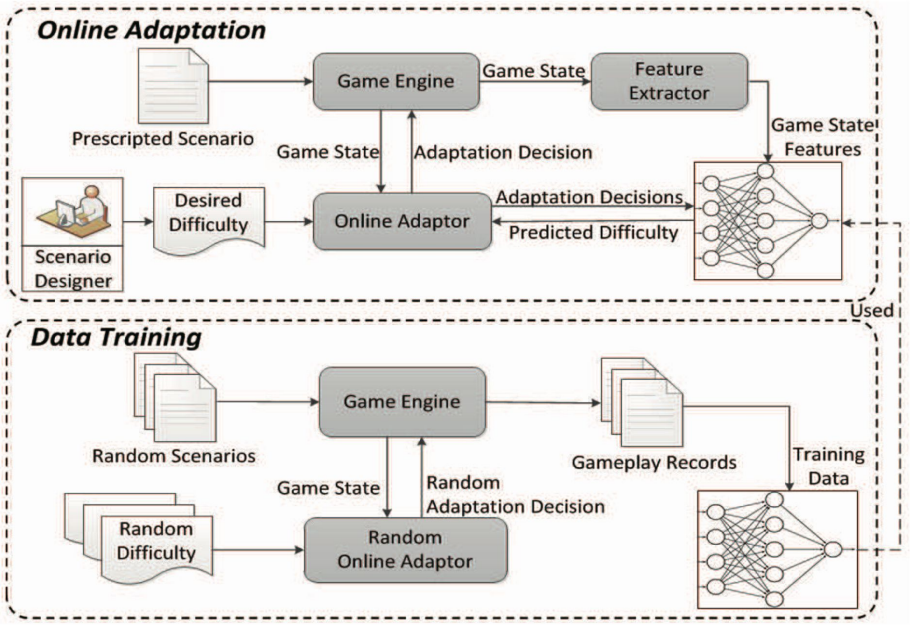
Predator-prey game

Results

- UCT can be applied in DDA in offline games by adjusting simulation time
- UCT can create data to train ANN
- ANNs trained with UCT created data *(trained with data generated with UCTs trained with different simulation times)* can be applied in DDA in offline & online games

(Li et al., 2010)

"A data-driven approach for online adaptation of game difficulty"



Dynamic Adaptation Framework

6. Upper Confidence Bound for Trees and Artificial Neural Networks

multi-agent serious training game

a data-driven approach for DDA of game scenarios

- by game designer specified difficulty preferences at different periods of time
- difficulty is inferred from dynamically measured performance value
- goal = fit the predicted experienced difficulty to the desired difficulty

Results:

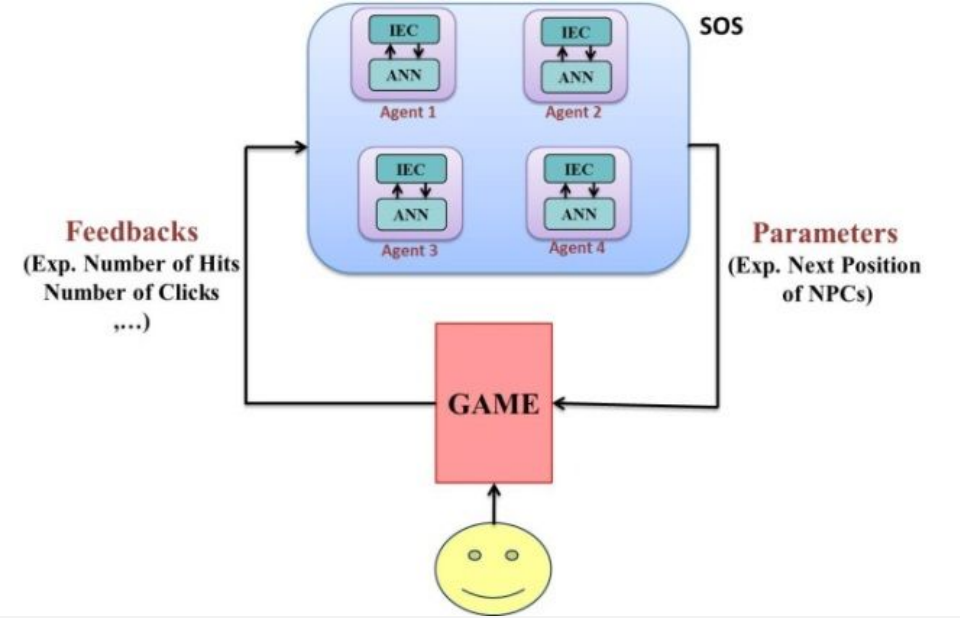
- Experiment demonstrated the efficacy and stability of the suggested approach.

(Yin et al., 2015)

7.

7. Self-Organizing System and Artificial Neural Networks

“Dynamic difficulty adjustment in games by using an interactive self-organizing architecture”



The proposed system

7. Self-Organizing System and Artificial Neural Networks

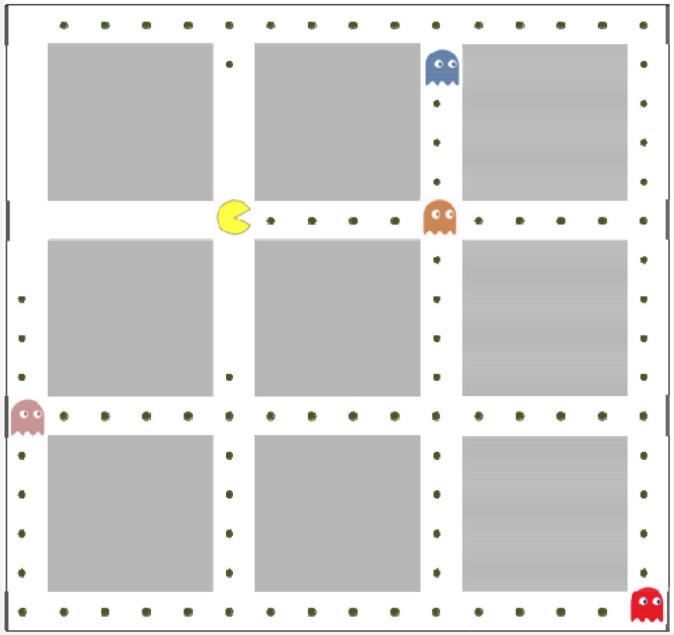
predator-prey game

Self-organizing system (SOS) of NPCs to adjust difficulty level of games

- Interactive evolutionary algorithm to evolve Non-Player Characters (NPCs)

(Ebrahimi & Akbarzadeh-T, 2014)

“Dynamic difficulty adjustment in games by using an interactive self-organizing architecture”



The Pacman testbed game.

7. Self-Organizing System and Artificial Neural Networks

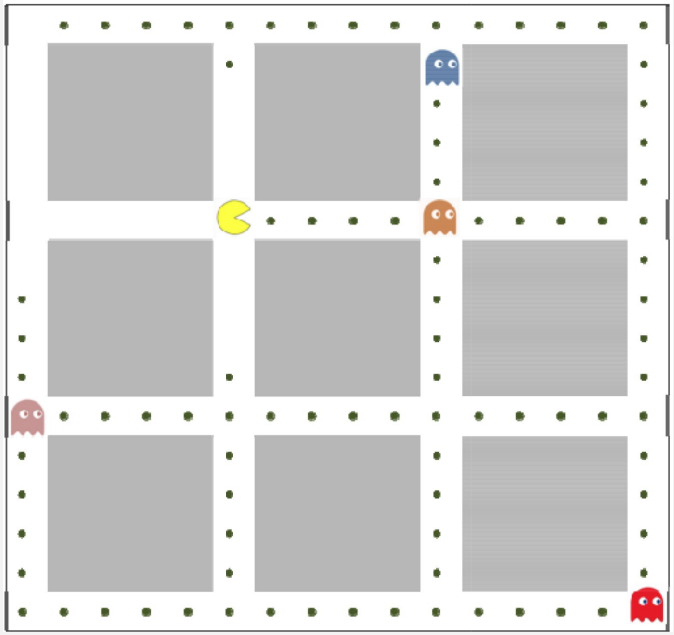
predator-prey game

Neuroevolutionary controller for each ghost to adapt to the different skill levels of the players

- movement based on environmental precepts

(Ebrahimi & Akbarzadeh-T, 2014)

“Dynamic difficulty adjustment in games by using an interactive self-organizing architecture”



The Pacman testbed game.

7. Self-Organizing System and Artificial Neural Networks

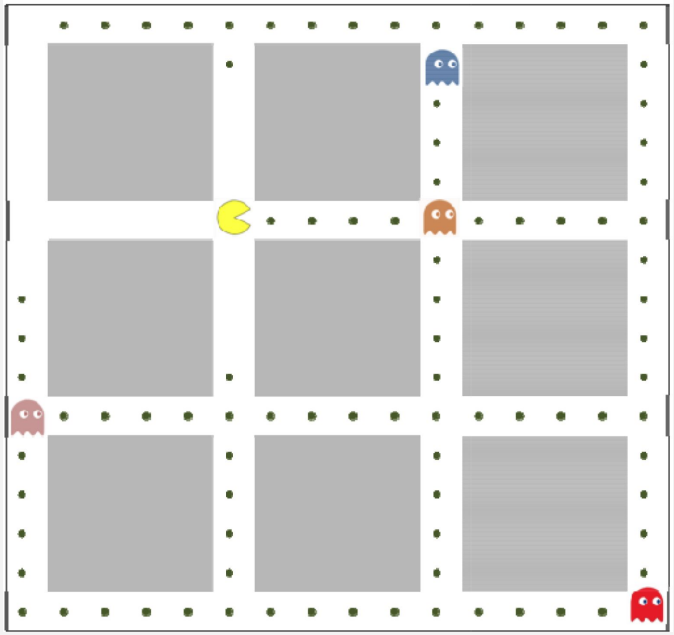
predator-prey game

Ghost controlled by fully connected feedforward neural network with a hidden layer

- default:
  - 4 inputs (*normalized*),
    - relative distance pacman x, y
    - relative distance closest ghost x, y
  - 4 outputs - movement options (*probabilities)*
  - 5 neurons in the hidden layer
  - connections [-1, 1]

(Ebrahimi & Akbarzadeh-T, 2014)

“Dynamic difficulty adjustment in games by using an interactive self-organizing architecture”



The Pacman testbed game.

7. Self-Organizing System and Artificial Neural Networks

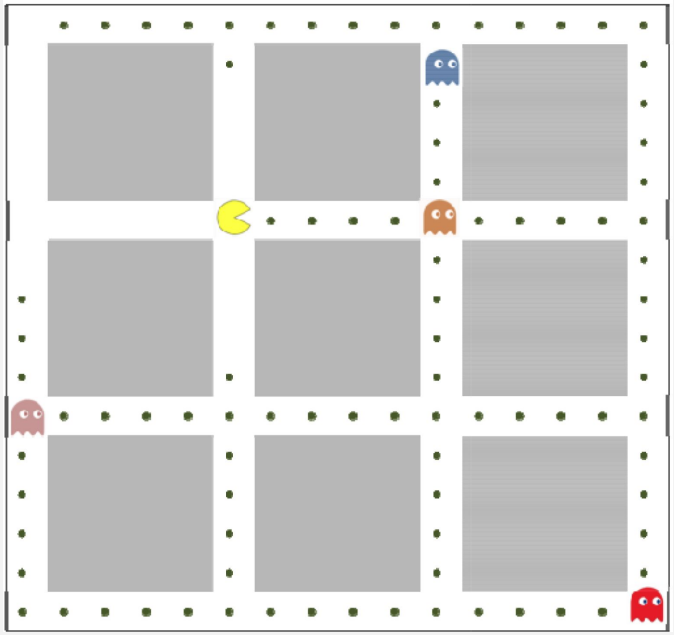
predator-prey game

Offline learning *(to create chromosomes that perform better than random chromosomes)*

- **Cooperative Coevolution** Algorithm to train offline
  - chromosomes subpopulation for every ghost
  - best performers in every subgroup are chosen as representatives
  - chromosomes of one subgroup are tested together with the representatives of the other subgroups

(Potter & de Jong, 2000, Ebrahimi & Akbarzadeh-T, 2014)

“Dynamic difficulty adjustment in games by using an interactive self-organizing architecture”



The Pacman testbed game.

7. Self-Organizing System and Artificial Neural Networks

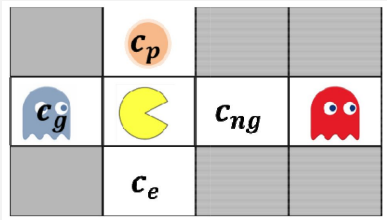
predator-prey game

Online learning

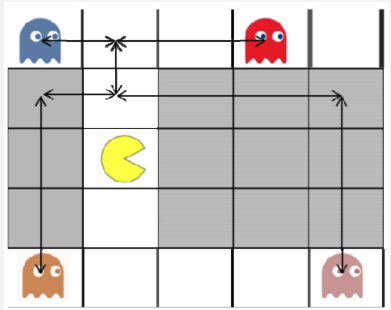
- Median controllers are loaded in ghosts at start of the game (*intermediate*)
- Interactive Evolutionary Computation (IEC) → fitness function replaced by human evaluation
  - assessed indirectly by reviewing player’s feedbacks
    calculated ideal key presses for distance traveled by Pacman, compared to actual key presses, occasions of key switches, wall hits, etc.

(Ebrahimi & Akbarzadeh-T, 2014)

"Dynamic difficulty adjustment in games by using an interactive self-organizing architecture"



Cost-based



Distance-based

7. Self-Organizing System and Artificial Neural Networks

predator-prey game

Four types of Pacman agents **varying intelligence levels**:

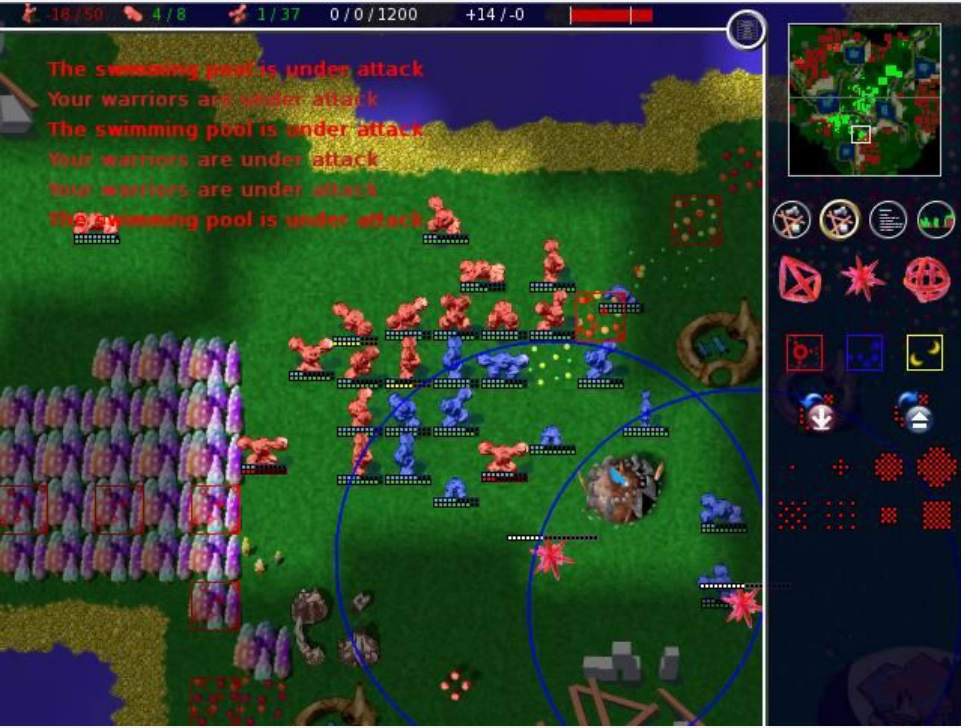
- cost-based
  considers state of neighbor cells
- nearest distance-based
  considers max distance to nearest ghost
- distance-based
  considers distance to all ghosts
- random

Results

- SOS can adapt itself with different level of skills

(Ebrahimi & Akbarzadeh-T, 2014)

“Real-time challenge balance in an RTS game using rtNEAT”



Globulation 2 test-bed game: screenshot during a combat.

7. Self-Organizing System and Artificial Neural Networks

real-time strategy (RTS) game

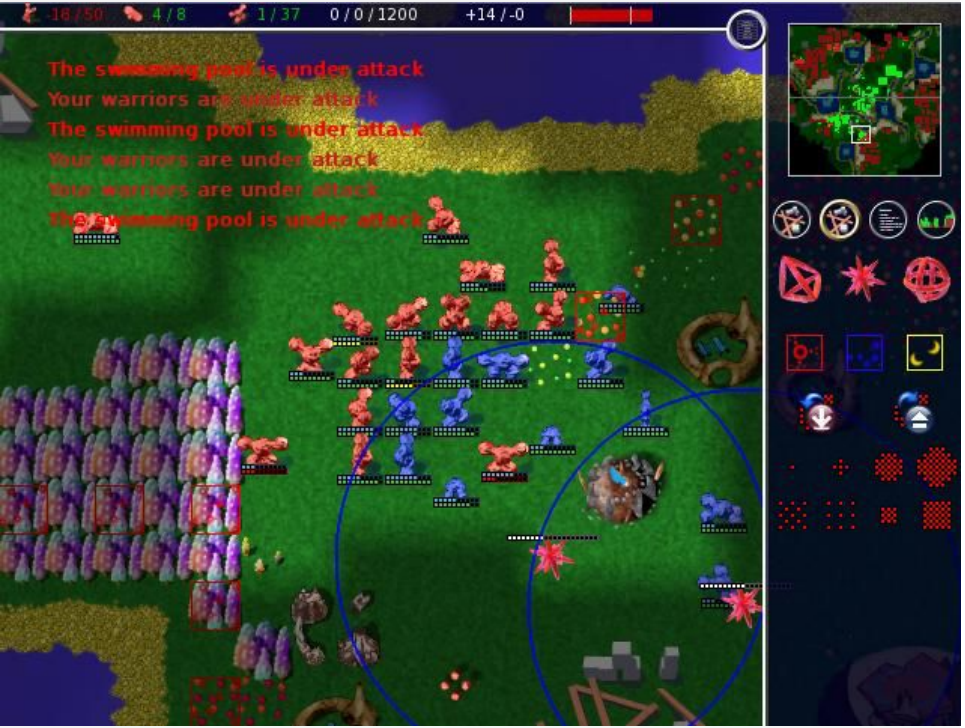
- Neuro-Evolution of Augmenting Topologies (NEAT)
- real-time Neat (rtNEAT)

… neuro-evolution methodologies are used to generate intelligent opponents

- NEAT adjusts both the connection weights and the topology of the network, thus it can:
  - modify value of the connection weights
  - add / remove nodes and connections

(Olesen et al., 2008)

“Real-time challenge balance in an RTS game using rtNEAT”



Globulation 2 test-bed game: screenshot during a combat.

7. Self-Organizing System and Artificial Neural Networks

real-time strategy (RTS) game

- Neuro-Evolution of Augmenting Topologies (NEAT)
- real-time Neat (rtNEAT)

… neuro-evolution methodologies are used to generate intelligent opponents

Results:

- effectiveness of NEAT and rtNEAT is indicated
- however, limitations for use in real-time strategy games

(Olesen et al., 2008)

8.

8. Affective Modeling Using EEG

“EEG-triggered dynamic difficulty adjustment for multiplayer games”

8. Affective Modeling Using EEG

third-person shooter game

Personal **excitement** level was monitored with an EEG[1] headset.

Experiment 1

- investigation of the **relationship** between **EEG signals** and **game events**

Results experiment 1

- **correlation** between the indicator for **short term excitement and game events**

(Stein et al., 2018)

“EEG-triggered dynamic difficulty adjustment for multiplayer games”

A screenshot from the customized version of the Boot Camp game.

8. Affective Modeling Using EEG

third-person shooter game

DDA chooses **modifications in favor of the weaker player** (*kill ratio*).

Experiment 2 - **comparison**:

- triggering of DDA modifications based on EEG - *decrease of excitement*
- triggering of DDA modifications based on typical heuristics *(elapsed duration, game status)*
- control game without DDA

Results experiment 2

- triggering of DDA modifications based on EEG increases player’s excitement and improving gaming experience

(Stein et al., 2018)

“Dynamic difficulty using brain metrics of workload”

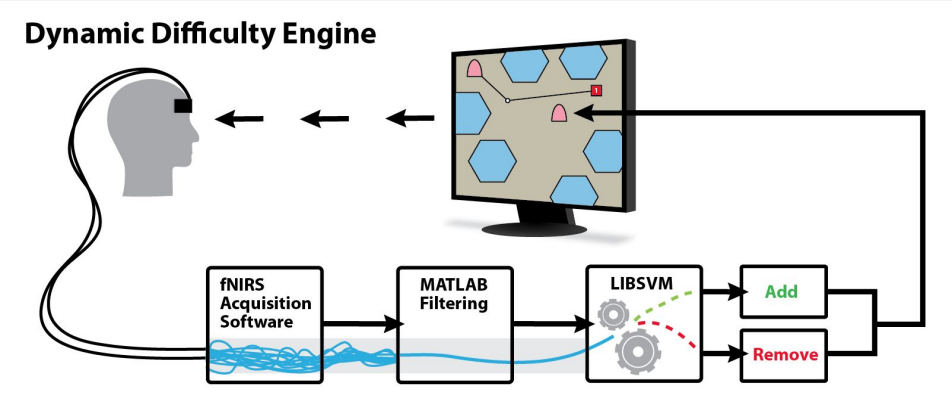


Diagram of the closed-loop dynamic difficulty adaptation engine. Raw signals acquired the fNIRS device are filtered, then used to classify user workload. When confident that the user is in a suboptimal state, UAVs are added / removed in order to provide the right amount of work.

8. Affective Modeling Using EEG

path planning task

Detection of extended **periods of boredom or overload** by collecting passive brain sensing information with functional near-infrared spectroscopy (fNIRS)

(Afergan et al., 2014)

“Dynamic difficulty using brain metrics of workload”

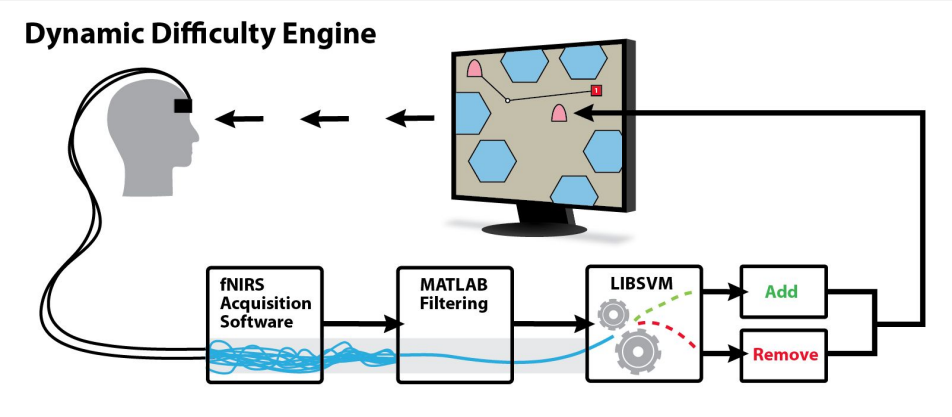


Diagram of the closed-loop dynamic difficulty adaptation engine. Raw signals acquired the fNIRS device are filtered, then used to classify user workload. When confident that the user is in a suboptimal state, UAVs are added / removed in order to provide the right amount of work.

8. Affective Modeling Using EEG

path planning task

Laboratory study:

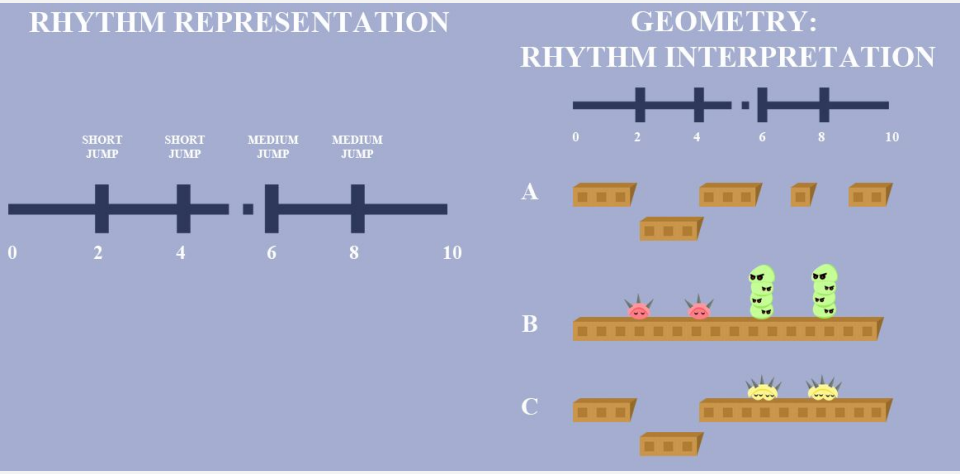
- participants performed path planning for multiple unmanned aerial vehicles (UAVs) in a simulation.
- based on participant’s state the task difficulty was varied by adding / removing UAVs

Results

- fNIRS brain sensing can be used to:
  - detect real-time task difficulty in real-time
  - construct an interface that improves user performance through DDA

(Afergan et al., 2014)

“Adaptable game experience based on player’s performance and EEG”



***Rhythm Representation**. Illustrated example of a rhythm and different geometry representations for a single rhythm.*

8. Affective Modeling Using EEG

2D platform game

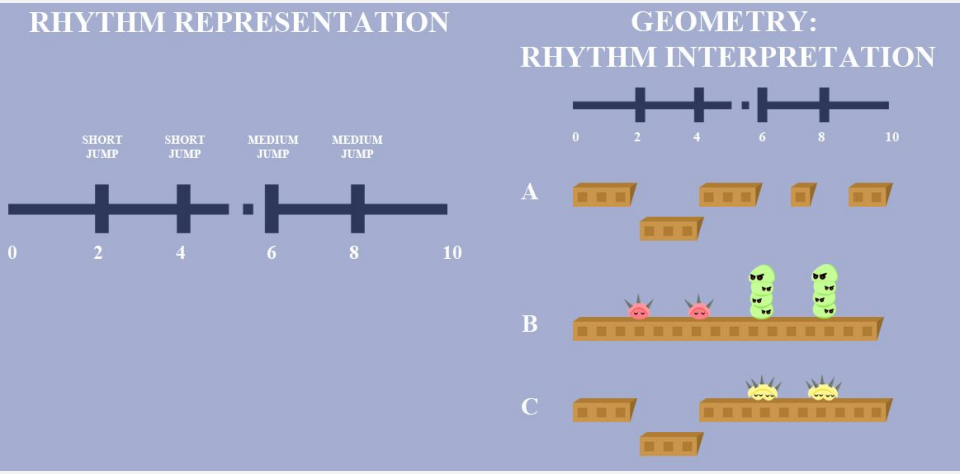
Procedural content development approach with the Rhythm-Group Theory to build varying platform levels.

DDA controlled by a combination of:

- player’s performance
- attention values based on EEG data

(Mikami & Kondo, 2017)

“Adaptable game experience based on player’s performance and EEG”



***Rhythm Representation**. Illustrated example of a rhythm and different geometry representations for a single rhythm.*

8. Affective Modeling Using EEG

2D platform game

Experiment: participants performed 5 automatically created levels based on their performance and estimated attention (EEG data).

Results

- The tested method is capable of adapting the difficulty of a level according to the player’s status, using values calculated in realtime.

(Mikami & Kondo, 2017)