Games AI

Lecture 11.1

Performance and Evaluation

Discuss:

In what ways can AI fail?

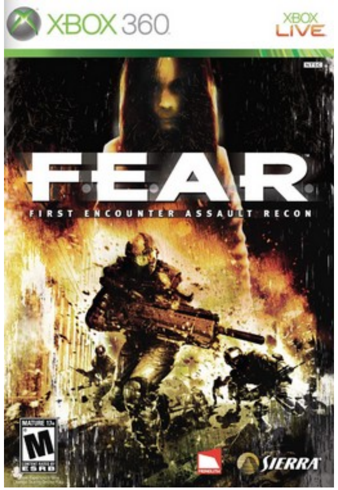
- What do AI systems need to achieve?
  - Specific task
  - Player experience
  - Support designers in doing their job
  - Satisfy development and runtime requirements

- How do we evaluate AI?
  - **Performance**
    - Resources consumed/required
    - Efficiency
  - **Qualitative**
    - Player experience
    - Critical quality


Performance
Untitled - pCARS Profiler
Tools
Open
Save
Close
Import
Export
File
Properties
Graphs
Indicators
Boards
Variance
Components
Time Axis
Single Color
Reset Zoom
Graph
Panels
Layout
Switch
View
Refresh
Lap
Session
Name
Time
Delta
Offset
Brands Hatch Indy
Aston Martin V8 Vantage GT4
19. 4. 2015 23:07 - Practice
Hockenheim Grand Prix
Ford Zakspeed Capri Group 5
19. 4. 2015 23:14 - Time Trial
19. 4. 2015 23:18 - Time Trial
19. 4. 2015 23:19 - Time Trial
General
Tire Wear
Speed [km/h]
Gear
Throttle [%]
Brake [%]
Variance
Tire Temp
TIRE TEMP FL
89
TIRE TEMP FR
81
TIRE TEMP RL
79
TIRE TEMP RR
74
Brake Temp
BRAKE TEMP FL
253
BRAKE TEMP FR
244
BRAKE TEMP RL
183
BRAKE TEMP RR
180
Engine RPM [rpm]
x1000
9202
95
0
3
Acceleration
Lat Accel [G]
Lng Accel [G]
Lap Timing
1:50.171
0:24.747
0:50.754
0:34.669
1:50.808
0:25.128
0:50.818
0:34.863
1:55.899
0:25.572
0:52.155
0:38.172
1:50.171
0:24.747
0:50.754
0:34.669
-0:05.729
2
1/1
3/3
Channels
Master
Overlay
Delta
Lap
Cursor
General
Input
Engine
Peaks
Track
Sectors
20 ms
Live
Paused

Performance

- Why is performance a big deal in **games**?
  - Technical quality is a selling point
    - Graphical fidelity
    - World size/scope
    - Realism
    - Number of units
  - User experience easily disrupted by e.g. frame rate



- Why is performance a big deal in **games AI**?
  - Algorithms tend to be performance intensive
  - Scales rapidly with number of agents, search depth, etc.
  - Often runs frequently (e.g. agent AI)
  - Can't be turned down like graphics settings
  - AI failures can be game-breaking

Discuss:

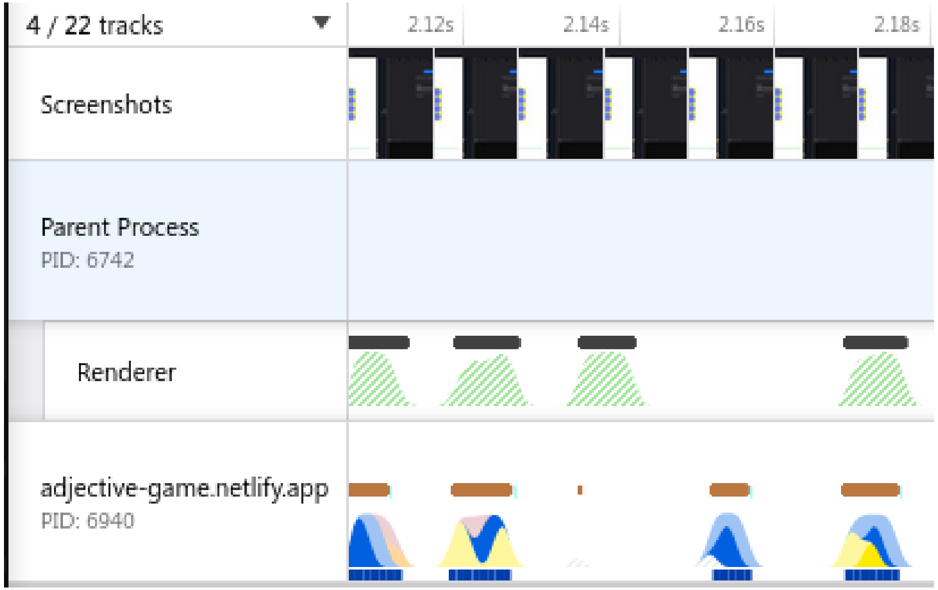
What contributes to performance?

- What contributes to performance?
  - Frame rate
  - CPU usage
  - Memory usage
  - GC calls
  - File size
  - … # draw calls
  - … network latency

“Premature optimisation is the root of all evil”

Donald Knuth

- Frame rate
  - Time to render a frame
  - Heavily influenced by graphics/rendering code
  - But all per-frame services need to complete as well
    - What AI needs to be run every frame?
  - Stuttering can be very obvious



- So…
  - Minimise per-frame work
  - Break out of loops if taking too long
  - Split tasks across multiple frames or use threads
  - How often does your AI really need to run?

- CPU usage
  - What hardware is needed
  - What else can be done at the same time?
  - Noise and heat
  - Battery life

- So…
  - Precompute (e.g. heuristics, content generation)
  - Do less work (run fewer enemies at once, fewer interactions, smaller models)
  - Lazy evaluation
  - Thread utilisaion

Performance

- Memory usage
  - Platform requirements/limitations
  - Memory fragmentation / read times
- So...
  - Careful use of data structures
  - **Data-oriented design**

Elemental: War of Magic by Stardock

- Modern computers:
  - CPUs are very fast
  - Memory is very large
  - Communication between two is bottleneck
- Problem:
  - CPUs constantly needing to wait to read memory
  - Waiting for main memory can take 100s of clock cycles

- **Solution:** Caching
  - Read ahead in memory and cache
    - Exploiting **locality of reference**: data needed is often close together
  - CPUs first check if data needed is in cache
    - **Cache hit**: the data needed is in cache
    - **Cache miss**: the data is not in cache and need to fetch from memory
  - Cache misses mean the CPU has to wait for the data
- **Problem**: If data needed is from different places in memory, there will be lots of cache misses

- Object-oriented programming
  - Organise source code around data types
  - Actual data could be anywhere in memory
    - Leading to more cache misses
- **Solution:** Physically group data together to improve cache performance
  - "Data-oriented design"
  - Parallel arrays

- **AoS**: Array of Structures

```
[
  {a: 1, b: 2, c: 3},
  {a: 10, b: 20, c: 30}
]
```

- **SoA**: Structure of Arrays

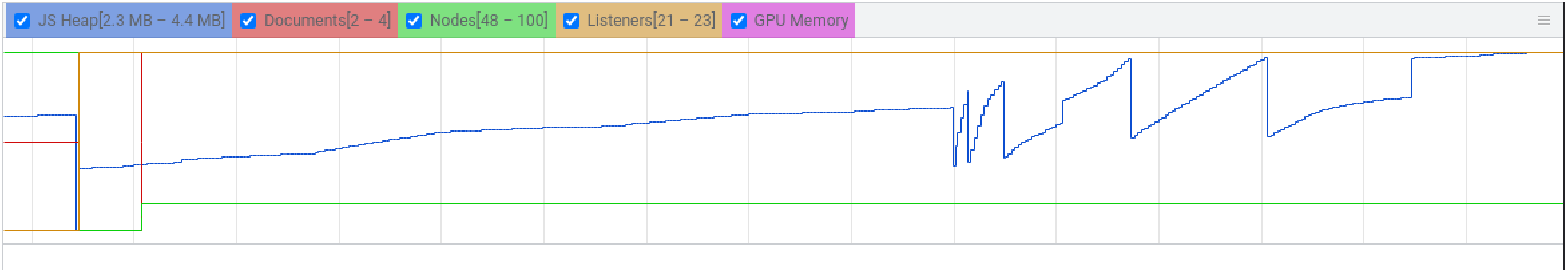
```
{
  a: [1, 10],
  b: [2, 20],
  c: [3, 30]
}
```

- Say you have a physics engine and want to accelerate objects due to gravity
  - Loop through array of object references
  - Look up each object (potentially anywhere)
  - Apply acceleration to velocity

- With parallel arrays
  - Loop through array of physics object **structs** (contiguous)
  - Apply acceleration to velocity

- **Warning:** Parallel arrays are more usually a seriously bad idea
  - There is a reason they are usually bad programming practice.
- Make sure you know what you are doing

- GC Calls
  - Garbage collection is CPU intensive and blocks main thread and can lead to frame stuttering

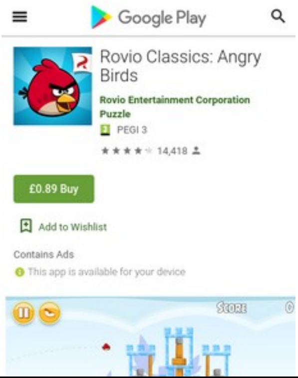


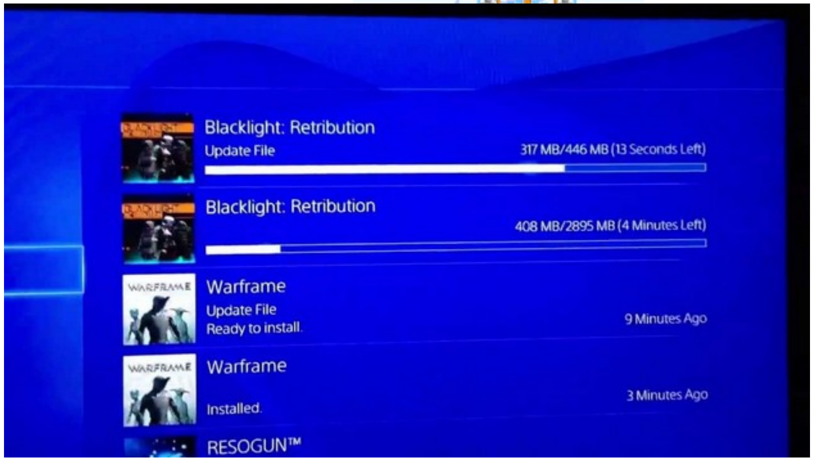
- GC in a nutshell:
  - Memory doesn't magically free itself
  - When you create an object (in e.g. C#, Java), a GC allocation is performed
  - The Garbage Collector keeps track of your object
  - When all references to your object are lost, the GC frees up the memory
  - Garbage collection is expensive

- Garbage collection can take a lot of thought to avoid
  - Avoid allocations to GC with a short lifetime (e.g. in main loop)
  - Allocations can be hard to spot
    - **new** operator
    - strings
    - resizing lists
    - API calls that return new objects or use them internally
    - Language features such as inline functions
    - Some methods for getting the current time

- Object pooling
  - Reuse objects so they don't need to be recreated
  - e.g. have a pool of sprite objects that you keep reusing
  - If you need 100s of nodes whenever doing pathfinding
    - Keep a pool of pathfinding nodes to reuse

- File size
  - Space consumed on device
  - Capacity of game media
  - Time to load/download



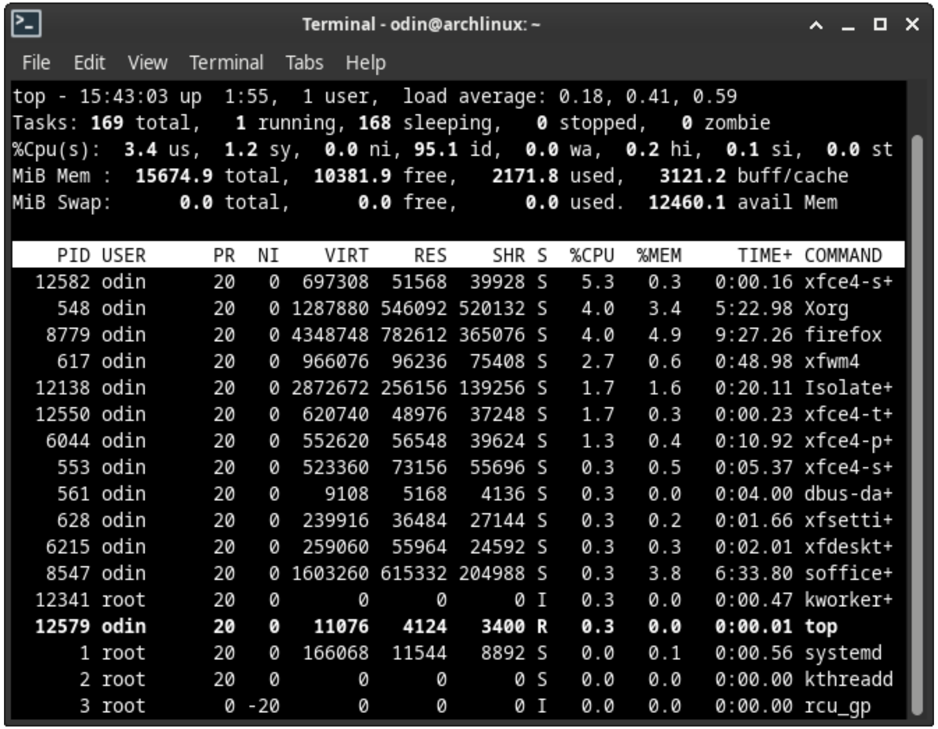


- So…
  - PCG to generate at runtime, procedural assets, or generate from seed
  - Stream game content

Evaluation
Technical
Untitled - pCARS Profiler
Tools
Open
Save
Close
Import
Export
Properties
Graphs
Indicators
Boards
Variance
Components
Time Axis
Single Color
Reset Zoom
Panels
Layout
Switch
Refresh
File
Components
Graph
View
Lap
Session
Name
Time
Delta
Offset
Brands Hatch Indy
Aston Martin V8 Vantage GT4
19. 4. 2015 23:07 - Practice
Hockenheim Grand Prix
Ford Zakspeed Capri Group 5
19. 4. 2015 23:14 - Time Trial
19. 4. 2015 23:18 - Time Trial
19. 4. 2015 23:19 - Time Trial
General
Tire Wear
Speed [km/h]
Engine RPM [rpm]
Gear
Throttle [%]
Brake [%]
Tire Temp
TIRE TEMP FL 89
TIRE TEMP FR 81
TIRE TEMP RL 79
TIRE TEMP RR 74
Brake Temp
BRAKE TEMP FL 253
BRAKE TEMP FR 244
BRAKE TEMP RL 183
BRAKE TEMP RR 180
Engine RPM [rpm]
x1000
9202
Acceleration
Lat Accel [G]
Lng Accel [G]
Channels
Master
Overlay
Delta
Cursor
Input
Engine
Peaks
Track
Sectors
Variance
Lap Timing
1:50.171
0:24.747
0:50.754
0:34.669
1:50.808
0:25.128
0:50.818
0:34.863
1:55.899
0:25.572
0:52.155
0:38.172
-0:05.729
2
1/1
3/3
Live
Paused

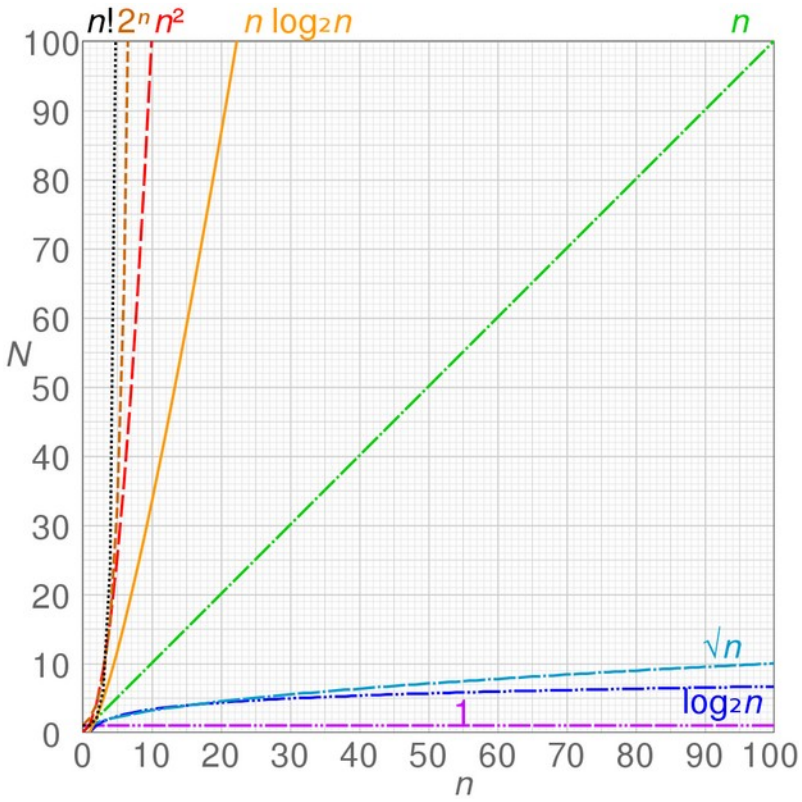
- How do we quantify the success of AI in a game?
  - Performance
  - Efficiency
  - Behavior

- Performance
  - Frame rate
  - CPU
  - Memory
- So just use … ?
  - FPS counter
  - `top` / Task manager
  - Profiler

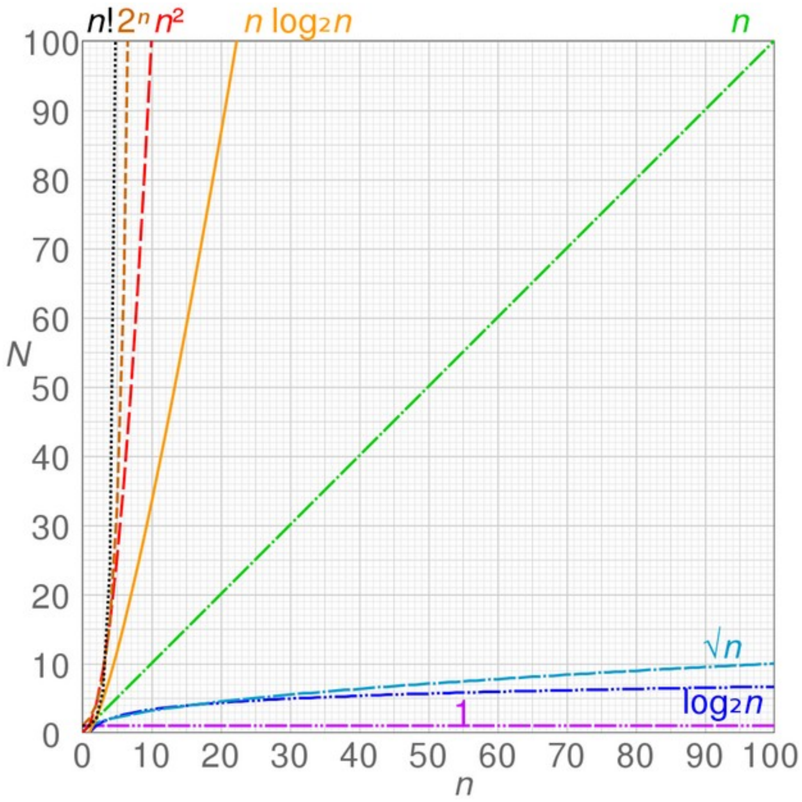


- Performance varies between
  - Platforms
  - Parts of the game
- Then what is the performance of an AI?
  - This is a solution to this for algorithms: **Asymptotic computational complexity**

- We care about how long algorithms take as their input gets bigger
  - Length of string
  - Size of array
  - width/height of a map
  - Number of things to sort
- Asymptotic complexity is an approach to calculate the rate at which time increases as input size increases



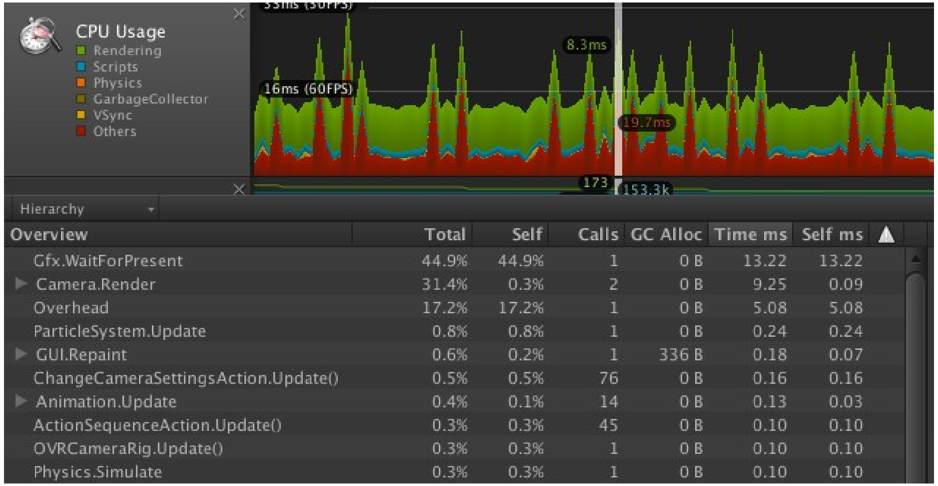
- A function f(x) might be described as
  - f(x) ∈ O(1) [constant]
  - f(x) ∈ O(log n) [logarithmic]
  - f(x) ∈ O(n) [linear]
  - f(x) ∈ O(n log n) [log linear]
  - f(x) ∈ O(n^2) [quadratic]
  - f(x) ∈ O(2^n) [exponential]
- for the best/average/worst case



- This is good for understanding algorithms, e.g.
  - Insertion Sort: $O(n_2)$
  - Quicksort: $O(n \log n)$
  - A*: $O(b^d)$
    - b = branching factor
    - d = depth
- But not for particular implementations or for the messy combinations often used in games

- So we need to (also) measure real-world performance
  - What platform to measure on?
    - Ideally: multiple target platforms
  - What to test?
    - Configurations in game
    - Finding the limits (how many agents, etc.)
- Disaggregate performance of AI specific code from game

- Measure performance
  - Use stand alone tools
  - Calculate performance data in-game and log
    - Capture remote logs
  - Use game engine-provided profiling tools



- Efficiency
  - How quickly does the AI find a solution?
    - How many steps / attempts / re-plans
  - What factors affect best/worst performance?

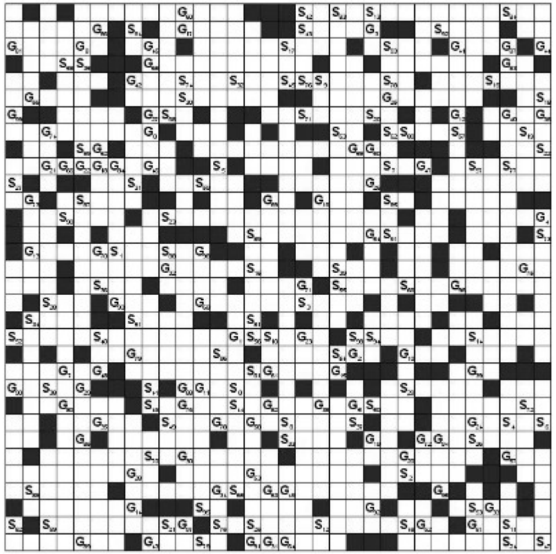


Figure 2: An example environment. Agents must navigate from S_i to G_i.

Proceedings of the First
Artificial Intelligence and Interactive
Digital Entertainment Conference
(AIIDE-2005)

Cooperative Pathfinding

David Silver

Department of Computing Science
University of Alberta
Edmonton, Canada T6G 2E8
silver@cs.ualberta.ca

Abstract

Cooperative Pathfinding is a multi-agent path planning problem where agents must find non-colliding routes to separate destinations, given full information about the routes of other agents. This paper presents three new algorithms for effi-

in order to ensure good beha
per introduces new algorithm
pathfinding more robustly and
challenging, real-time enviror
Real-Time Strategy games

- In pathfinding its common to use A* with local repair
  - Plan routes independantly
  - If agents are about to run into each other they re-plan (and again, and again)
- In dense maps many replans may be required
- Cooperative pathfinding approaches are generally slower, but require fewer replans.

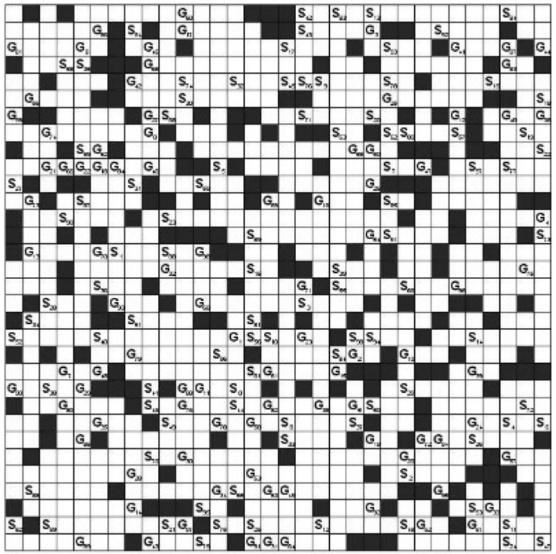


Figure 2: An example environment. Agents must navigate from S_i to G_i.

Proceedings of the First
Artificial Intelligence and Interactive
Digital Entertainment Conference
(AIIDE-2005)

Cooperative Pathfinding

David Silver

Department of Computing Science
University of Alberta
Edmonton, Canada T6G 2E8
silver@cs.ualberta.ca

Abstract

Cooperative Pathfinding is a multi-agent path planning problem where agents must find non-colliding routes to separate destinations, given full information about the routes of other agents. This paper presents three new algorithms for effi-

in order to ensure good beha
per introduces new algorithm
pathfinding more robustly and
challenging, real-time enviror
Real-Time Strategy games

- Generate and test methods
  - What proportion of generated content is usable?
  - How many iterations/generations are required?
- Machine Learning
  - How big a model do you need for good results?
  - Train for how many epochs?

- Computer Science is generally interested in finding efficient algorithms
  - Find fancy algorithms
  - Implementation is secondary
- Game developers are generally interested what works now
  - Tried and tested algorithms
  - Efficient implementations

Technical Evaluation

- Behavior
  - How many errors/crashes/misses
  - How often is each behavior used?
  - Win rate/average score
  - Generative space
    - Typical examples
    - What are the boundaries?

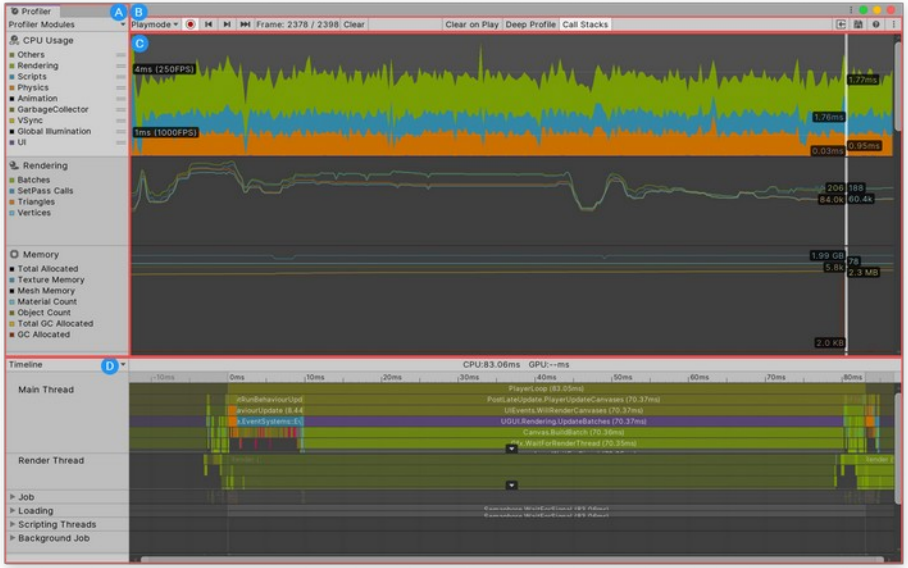

Untitled - pCARS Profiler
Measurement
Tools
Open
Save
Close
Import
Export
File
Properties
Graphs
Indicators
Boards
Variance
Components
Time Axis
Single Color
Reset Zoom
Graph
Panels
Layout
Switch
View
Refresh
General
Tire Wear
Lap
Session
Name
Time
Delta
Offset
Brands Hatch Indy
Aston Martin V8 Vantage GT4
Hockenheim Grand Prix
Ford Zakspeed Capri Group 5
Tire Temp
TIRE TEMP FL
89
TIRE TEMP FR
81
TIRE TEMP RL
79
TIRE TEMP RR
74
Brake Temp
BRAKE TEMP FL
253
BRAKE TEMP FR
244
BRAKE TEMP RL
183
BRAKE TEMP RR
180
Engine RPM [rpm]
9202
Gear
Acceleration
Channels
Track
Sectors
Variance
Lap Timing
1:50.171
0:24.747
0:50.754
0:34.669
1:50.808
0:25.128
0:50.818
0:34.863
1:55.899
0:25.572
0:52.155
0:38.172
-0:05.729
2
1/1
3/3

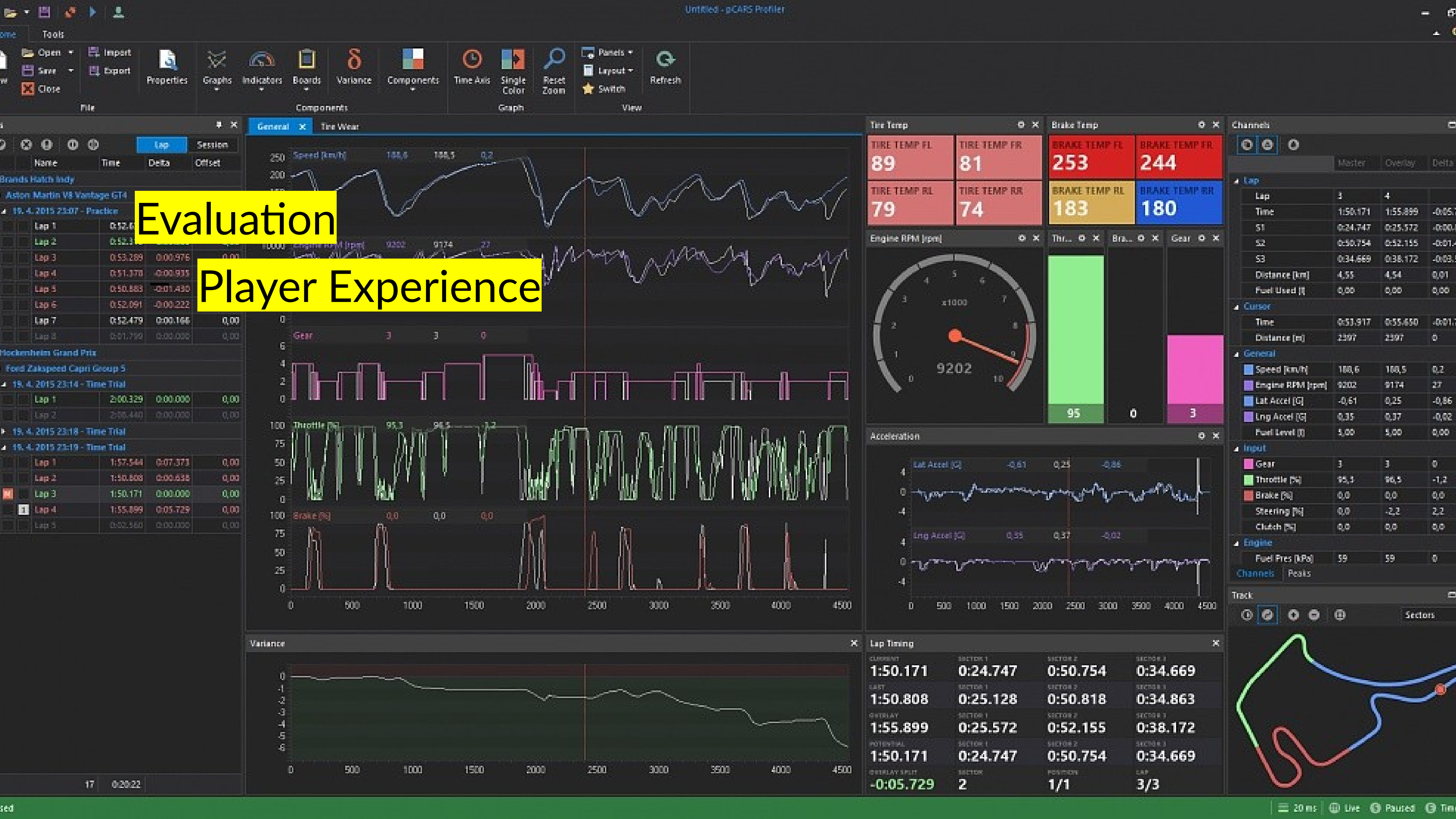
- How to measure
  - Manual instrumenting
  - Logging/telemetry
  - External tools/Profilers

- Instrumenting
  - Get current time (efficiently!)
    - Time.realtimeSinceStartup()
  - Instrument sections of code e.g. main loop, pathfinding algorithm

- Telemetry
  - "Remote measurement"
  - Get data out of your game
    - Log file
    - Database
      - e.g. send data to RESTful database with HTTP request

- Profiling Performance
  - Unity Profiler
- **Task:** Open Unity, go to Window > Analysis > Profiler and record a section of gameplay to analyse



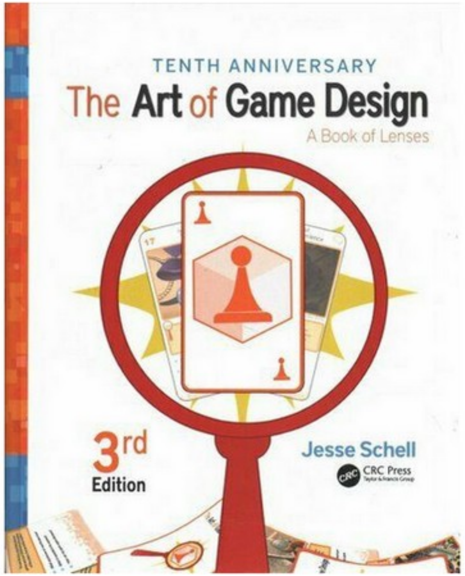


Evaluation
Player Experience
Untitled - pCARS Profiler
Tools
Open
Save
Close
Import
Export
Properties
Graphs
Indicators
Boards
Variance
Components
Time Axis
Single Color
Reset Zoom
Panels
Layout
Switch
Refresh
File
Graph
View
General
Tire Wear
Lap
Session
Name
Time
Delta
Offset
Brands Hatch Indy
Aston Martin V8 Vantage GT4
Hockenheim Grand Prix
Ford Zakspeed Capri Group 5
Tire Temp
Brake Temp
Channels
Engine RPM [rpm]
Acceleration
Variance
Lap Timing
Track
Sectors

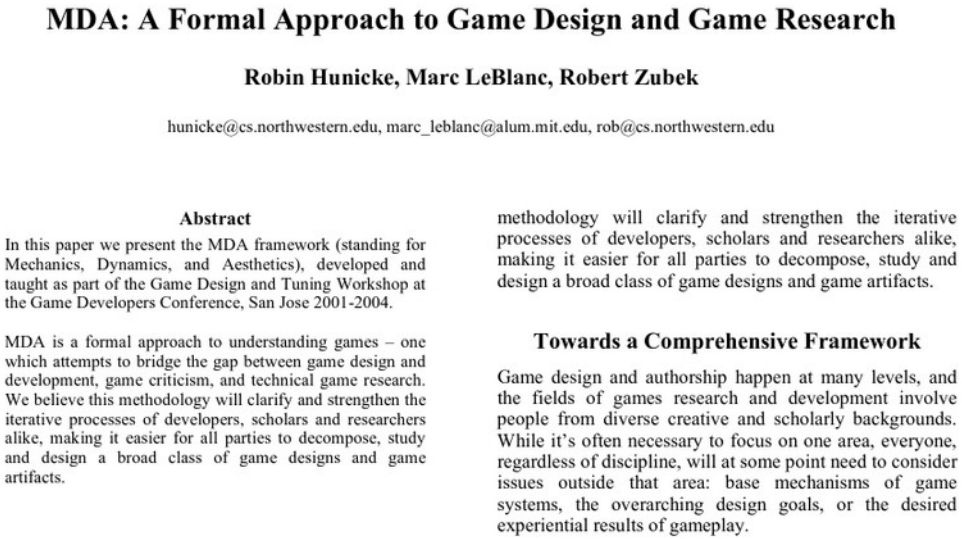
- Player experience of what?
  - Downloading/installing/launching
  - First play?
  - Tutorial?
  - Particular level?
  - Particular (new) feature?
  - Why users quit? – churn
- AI:
  - Interaction with NPCs
  - Quality of PCG
  - Balance of game director

- Methodologies
  1) Game designer’s knowledge
  2) User testing
  3) Models of the player

- Game design models and lenses
  - Balance
    - Game theory
    - Probability
  - Economy
  - Meaningful choices
  - MDA



- **Mechanics**
  - AI implementation
- **Dynamics**
  - In-world agent behavior and interactions
- **Aesthetics**
  - Emotional responses in the player



MDA: A Formal Approach to Game Design and Game Research

Robin Hunicke, Marc LeBlanc, Robert Zubek

hunicke@cs.northwestern.edu, marc_leblanc@alum.mit.edu, rob@cs.northwestern.edu

Abstract

In this paper we present the MDA framework (standing for Mechanics, Dynamics, and Aesthetics), developed and taught as part of the Game Design and Tuning Workshop at the Game Developers Conference, San Jose 2001-2004.

MDA is a formal approach to understanding games – one which attempts to bridge the gap between game design and development, game criticism, and technical game research. We believe this methodology will clarify and strengthen the iterative processes of developers, scholars and researchers alike, making it easier for all parties to decompose, study and design a broad class of game designs and game artifacts.

methodology will clarify and strengthen the iterative processes of developers, scholars and researchers alike, making it easier for all parties to decompose, study and design a broad class of game designs and game artifacts.

Towards a Comprehensive Framework

Game design and authorship happen at many levels, and the fields of games research and development involve people from diverse creative and scholarly backgrounds. While it's often necessary to focus on one area, everyone, regardless of discipline, will at some point need to consider issues outside that area: base mechanisms of game systems, the overarching design goals, or the desired experiential results of gameplay.

- **Mechanics**
  - Commitment bonus (utility-based AI)
- **Dynamics**
  - Consistency in action
- **Aesthetics**
  - AI appears decisive and intentional

  … which makes the player feel …

- **Mechanics**
  - A*
- **Dynamics**
  - Shortest path navigation
- **Aesthetics**
  - AI appears aware of surroundings
  - AI seems robotic

  … which makes the player feel …

- User testing
  - Game reviews
  - Player experience questionnaires
    - PENS, GEQ, IMI, UPEQ…
  - Speak-aloud
  - Interviews

UPEQ: Ubisoft Perceived Experience Questionnaire

A self-determination evaluation tool for video games

Ahmad Azadvar
Massive Entertainment Sweden AB
SE-203 14 Malmö
Sweden
ahmad.azadvar@massive.se

Alessandro Canossa
Massive Entertainment Sweden AB
SE-203 14 Malmö
Sweden
alessandro.canossa@massive.se

Abstract

In order to appeal to a growing market, game developers are offering a wide variety of activities. It is becoming necessary to understand which psychological need each activity caters for. The purpose of this paper is to demonstrate the development and evaluation of an instrument to assess which basic psychological needs are satisfied by different video games. This work is part of a growing effort in HCI to develop surveys able to capture subtle nuances of the player experience. This model, UPEQ, was developed by transforming a self-determination theory questionnaire into a video game specific survey. UPEQ consists of three subscale of Autonomy, Competence and Relatedness, which, measures of sense of transportation to the game as well as enjoyment of and engagement with the game. Regression with in-game behavior of players tracked by game engine also confirmed that each subscale of UPEQ independently predicts playtime, money spent on the game and playing as a group.

CCS Concepts

CCS → Human-centered computing → Human computer interaction (HCI) → HCI design and evaluation methods → User models.

KEYWORDS

The Convergence of Player Experience Questionnaires

Alena Denisova
Department of Computer Science, University of York
York, United Kingdom
ad595@york.ac.uk

A. Imran Nordin
Institute of Visual Informatics, The National University of Malaysia, UKM Bangi
Selangor, Malaysia
imran.nordin@gmail.com

Paul Cairns
Department of Computer Science, University of York
York, United Kingdom
paul.cairns@york.ac.uk

ABSTRACT

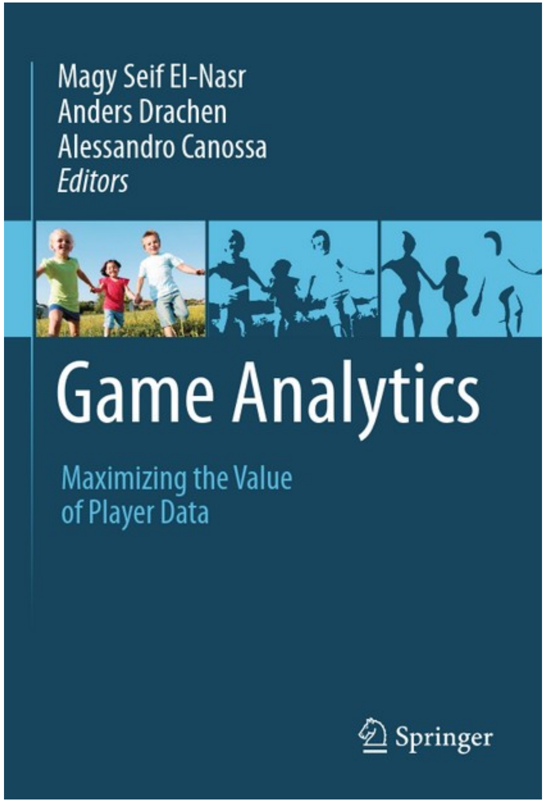
Player experience is an important field of digital games research to understand how games influence players. A common way to directly measure players' reported experiences is through questionnaires. However, the large number of questionnaires currently in use introduces several challenges both in terms of selecting suitable measures and comparing results across studies. In this paper, we review some of the most widely known and used questionnaires and focus on the immersive experience questionnaire (IEQ), the game engagement questionnaire (GEQ), and the player experience of need sat-

experience under consideration, while being relatively easy to deploy [1]. Like the more objective measures, the use of questionnaires ensures consistency and uniformity of collected data, because the same specific aspects are considered by all participants in all studies.

There are, however, a few drawbacks of using questionnaires to measure player experience. Nordin et al. [12] named the challenges researchers face when looking for the most appropriate questionnaire. Amongst these, they note, is the ability to persuade participants to treat the questionnaires seriously,

- Pros
  - Data from real players
- Cons
  - Expensive
  - Time consuming (slows development)

- Game Analytics
  - Extract data from play using **telemetry**
  - Process raw telemetry data into **game metrics**
  - Use metrics as a source of **business intelligence**



- Raw data
  - Start/end time
  - Level completion time
  - Moves/time per level
  - Location of player

- Game Metrics
  - Churn rate
  - Difficulty
  - % level explored

- Player modeling
  - Clustering
  - Prediction
    - Behavior
    - Experience

- Models of the player
  - How should we understand the behavior / motivations / experiences of the player?
  - What factors influence the player?
  - How does our AI support or undermine these?

- Self-Determination Theory
  - Autonomy
  - Competence
  - Relatedness

Motiv Emot (2006) 30:347–363
DOI 10.1007/s11031-006-9051-8

ORIGINAL PAPER

The Motivational Pull of Video Games: A Self-Determination Theory Approach

Richard M. Ryan · C. Scott Rigby · Andrew Przybylski

Published online: 29 November 2006

Abstract Four studies apply self-determination theory (SDT; Ryan & Deci, 2000) in investigating motivation for computer game play, and the effects of game play on well-being. Studies 1–3 examine individuals playing 1, 2 and 4 games, respectively and show that perceived in-game autonomy and competence are associated with game enjoyment, preferences, and changes in well-being pre- to post-play.

power, coupled with the integration of the Internet into mainstream society, has given birth to numerous gaming environments and "virtual worlds," that are increasingly complex, immersive, engaging, and enabling of a wide range of activities, goals, and social behavior.

Of particular relevance to the research we present in this article are those computer environments associated with

- Flow

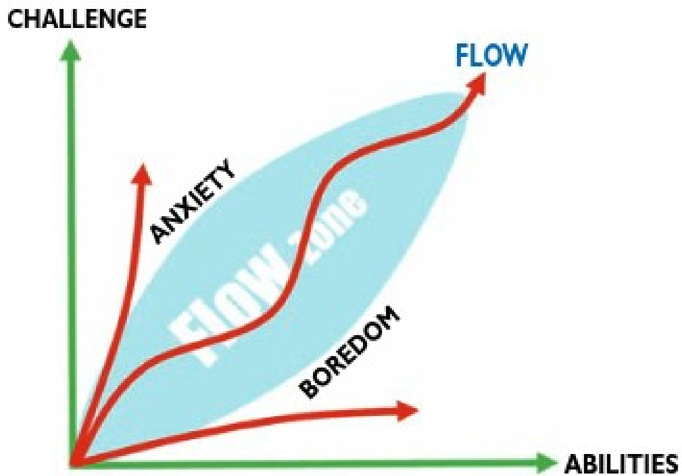


Figure 1. Flow Zone factors.

Viewpoint | Jenova Chen

Flow in Games (and Everything Else)

A well-designed game transports its players to their personal Flow Zones, delivering genuine feelings of pleasure and happiness.

Have you ever enjoyed an interactive experience, even as it was ignored by others? Why does a particular experience have broader appeal than others? Answers depend

ers can apply in their own designs. To do so, however, they must first know what exactly happiness is made of.

In the mid-1970s, in an attempt to explain happiness, Mihaly Csikszentmihalyi, a professor of psychology at the Claremont Graduate University,

Summary

Untitled - pCARS Profiler
Tools
Open
Save
Close
Import
Export
Properties
Graphs
Indicators
Boards
Variance
Components
Time Axis
Single Color
Reset Zoom
Panels
Layout
Switch
Refresh
File
Components
Graph
View
General
Tire Wear
Lap
Session
Name
Time
Delta
Offset
Brands Hatch Indy
Aston Martin V8 Vantage GT4
Hockenheim Grand Prix
Ford Zakspeed Capri Group 5
Speed [km/h]
Engine RPM [rpm]
Gear
Throttle [%]
Brake [%]
Tire Temp
TIRE TEMP FL 89
TIRE TEMP FR 81
TIRE TEMP RL 79
TIRE TEMP RR 74
Brake Temp
BRAKE TEMP FL 253
BRAKE TEMP FR 244
BRAKE TEMP RL 183
BRAKE TEMP RR 180
Engine RPM [rpm]
9202
Acceleration
Lat Accel [G]
Lng Accel [G]
Channels
Master
Overlay
Delta
Cursor
General
Input
Engine
Peaks
Track
Sectors
Variance
Lap Timing
1:50.171
0:24.747
0:50.754
0:34.669
1:50.808
0:25.128
0:50.818
0:34.863
1:55.899
0:25.572
0:52.155
0:38.172
-0:05.729
2
1/1
3/3
Live
Paused

- Performance
  - Optimisation
  - Measurement
- Player Experience
  - User testing
  - Game analytics
  - Player models