POLIMI tu

# Computational Intelligence in Games

Daniele Loiacono and Mike Preuss

GECCO, July 7-11, 2012, Philadelphia, USA

# Presenters

### Daniele Loiacono

He is assistant professor at the Department of Electronics and Information of Politecnico di Milano, where, in 2008, he received his Ph.D. in Computer Engineering. His research interests include machine learning, evolutionary computation, and computational intelligence in games. Since 2008, Daniele Loiacono has been organizing several scientific competitions at major conferences including GECCO, CEC and CIG. In 2009 he was local co-chair of the IEEE Symposium on Computational Intelligence and he was Competitions Chair of GECCO 2012

### Mike Preuss

Research Associate at the Computer Science Department, TU Dortmund, Germany, where he also received his Diploma degree in 1998. His research interests focus on the field of evolutionary algorithms for real-valued problems, namely on multimodal and multiobjective niching and the experimental methodology for (non-deterministic) optimization algorithms. He is currently working on the adaptability and applicability of computational intelligence techniques for various engineering domains and computer games, pushing forward modern approaches of experimental analysis as the Exploratory Landscape Analysis (ELA) and innovative uses of surrogate models.

# Introduction

**Part I**

- ❑ What is Computational Intelligence and Games about?
- ❑ What are the opportunities for Evolutionary Computation methods?
- ❑ The industry connection

**Part II**

- ❑ Games as testbed
- ❑ Developing better games
- ❑ Developing *innovative* games

**Part III**

- ❑ Competitions and available software

# CIG: An Overview

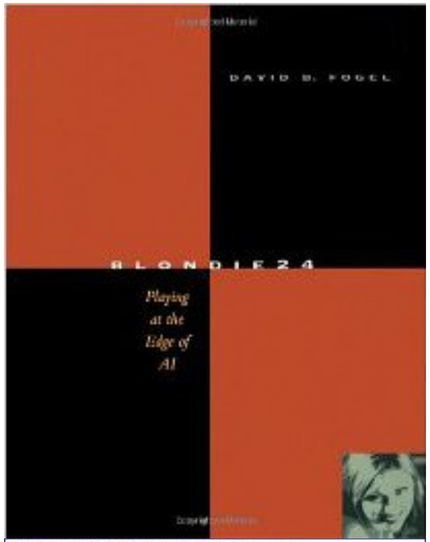
# Beginnings I: gaming

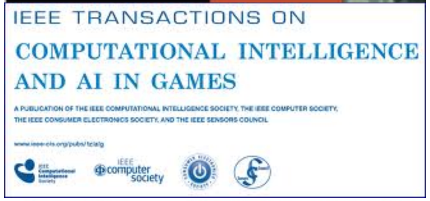
| | |
|---|---|
| 3000 BC | Dice, Senet |
| 2300 BC | Go |
| 500 AD | Chess |
| ca. 1600 | Modern sports games |
| ca. 1800 | Poker, Bridge |
| 1871 | Pinball |
| ca. 1935 | Monopoly, Scrabble |
| 1943 | Game theory beginnings |
| 1959 | Diplomacy |

# Beginnings II: computer gaming

| | |
|---|---|
| 1961 | Spacewar! - first computer video game |
| 1971 | Galaxy Game - first arcade video game |
| 1972 | Magnavox Odyssey console |
| 1973 | Game theory: Evolutionary stable strategies |
| 1978-81 | Space Invaders, PacMan, Donkey Kong |
| 1983 | I, Robot - first commercial 3D video game |
| 1992 | Wolfenstein 3D - popularization of FPS (first person shooters) |
| 1997 | Ultima Online - first massive multiplayer online (MMO) game |
| 1997 | Deep Blue beats Garry Kasparov |
| 1999 | Blondie24: Playing Checkers by means of CI |
| 2006 | Wii |
| 2008 | Checkers solved |

# Beginnings III: A field forming




IEEE TRANSACTIONS ON

COMPUTATIONAL INTELLIGENCE AND AI IN GAMES

A PUBLICATION OF THE IEEE COMPUTATIONAL INTELLIGENCE SOCIETY, THE IEEE COMPUTER SOCIETY, THE IEEE CONSUMER ELECTRONICS SOCIETY, AND THE IEEE SENSORS COUNCIL

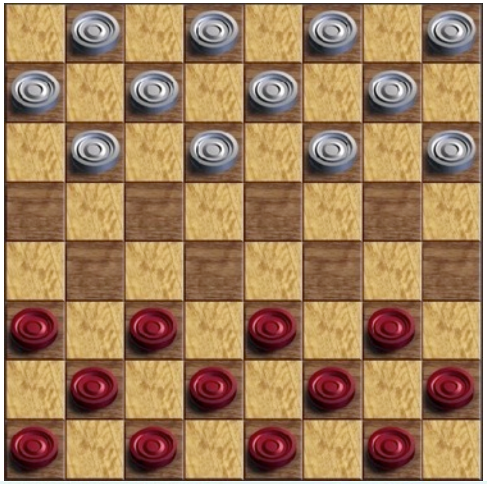
www.ieee-cis.org/pubs/tciaig

- 1999: Blondie24, Learning checkers with CI and human players
- GECCO before 2005: max 2 papers/year
- 2005 first Computational Intelligence in Games (CIG) conference
- GECCO after 2008: around 10 papers/year
- IEEE TCIAIG Journal (Transactions on CI and Artificial Intelligence in Games) since 2009
- EvoGames track in Evo* since 2009
- 2012: first Dagstuhl seminar on AI and CI in Games
- Many “neighbor” conferences, etc. AIIDE, FDG, gameai conf. (not strictly CI, but CI welcome)
- General approach is target oriented, not technique oriented

# Computer Games: trends and problems

- ❑ About 40 years of development:
  - From simplest graphics to virtual reality
  - Games use the current hardware potential
  - Graphis still dominate public perception of games, AI unimportant
  - Game production consists of: game design, storyline design, game mechanics, level design/content creation, character design, physics, playtesting etc.
  - Often teams of 50+ people for several years
- ❑ Problems:
  - Complex game realities require complex AI behavior to achieve Believability
  - Complex game worlds need huge effort to create content

# Believability

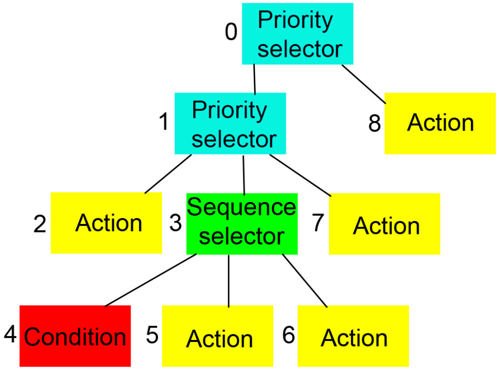


- Board game AI already quite good
  - Deep blue (IBM) beats Kasparov 1997
  - Checkers solved in 2008 (Schaffer)
  - Monte Carlo Tree Search (MCTS) has huge impact on e.g. Go AI
- More challenges in other games:
  - Believable appearance and behavior of all game components
  - NPC are a major problem (therefore MMOG)
  - Should act intelligently (or create this impression) and react appropriately
  - Must not reveal their identity by means of stupid mistakes (e.g. behavior loops)

# Authenticity

- Some standard game AI problems example: Gothic 3
  - Path finding ineffizient, unrealistic paths
  - Interaction of game ai and physics engine: mimics, gestures, movements
  - Camera movement (e.g. following head but not entering the same room)
  - Again: Repetitions (game AI always reacts in the same way)
- Problem is tackled by modularization: Middleware
  - Specialized physics engines
  - Complex character modelling e.g. with EkiOne (emotions)
  - Difficulty: We may only use about 10% CPU-time for the whole AI

# Standard game AI approaches



- ❑ Game industry prefers well known techniques
  - Scripting
  - Rule based systems
  - Finite state machines (also hierarchical)
  - New: behavior trees
- ❑ Industry cautious concerning dynamics and non-determinism
  - What will we get?
  - How can we control game flow?
- ❑ Current development very dynamic, e.g. look at: http://aigamedev.com/
- ❑ However, most current CIG research goes unnoticed by industry

# Research trees

- Research approaches games (mainly) from 3 directions
  - Specialized algorithms: Exact algorithms or heuristics, e.g. applied to path finding (A*)
  - The ’classic’ (deterministic) AI approach: General game playing (game description language GDL), tree search, also support vector machines (SVM) and reinforcement learning, strong in board games
  - Computational Intelligence (CI): Evolutionary algorithms, fuzzy logic, artificial neural networks, swarm intelligence etc., often applied for complex black-box controllers, analysing data

- However, there are overlaps. . .

# Why shall we apply CI (evolutionary) methods to games?

- Contrary to board games,
  - Game trees often not applicable
  - Incomplete information
  - Concurrency: During planning phase, the game situation changes
  - Quantifying a game situation is not trivial

→ Good and fast approximations are needed

- Evolutionary Optimization is
  - Versatile, flexible, still works (somehow)
  - Copes with noise and strange search spaces
  - Can be asked to deliver a result at any time

# What is the use of CI (evolutionary) methods in games research?

Lucas/Kendall 2006 "Evolutionary Computation and Games" (IEEE Computational Intelligence Magazine)

1. Good testbed to apply our methods
2. Do things in a better way
3. Do things we (or they) could not do before

# Games as testbeds

## Why games are good testbeds?

Unbiased

More complex than academic benchmarks

Challenging requirements (e.g., real-time)

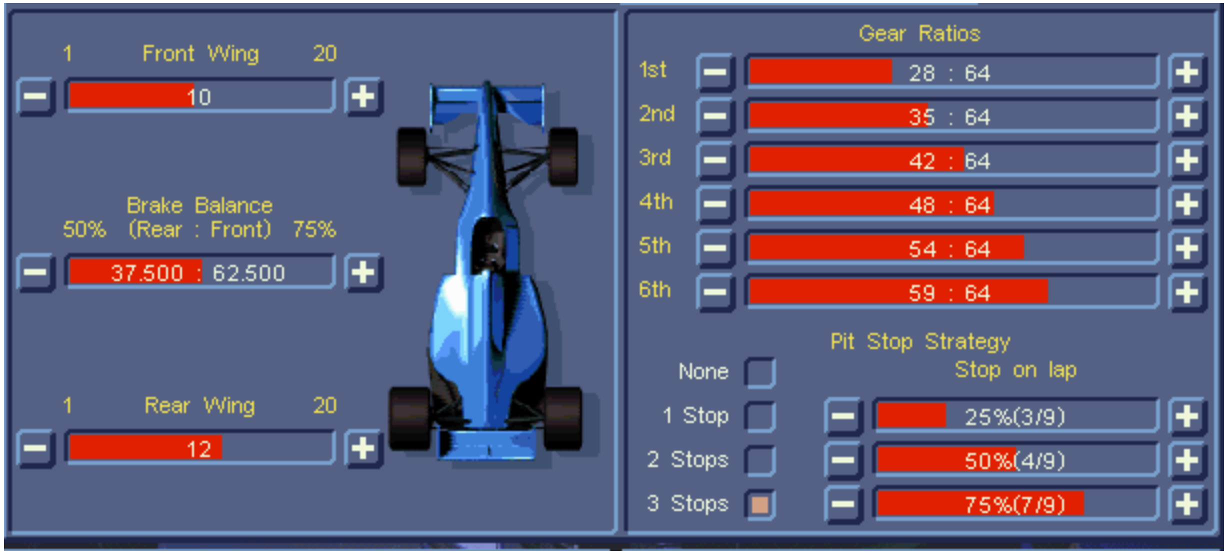
Cheaper than *real-world* problems

Human interaction

# Car Setup Optimization Competition

# Car Setup Optimization: Overview

- The goal is finding the best car setup on three unknown tracks
- Challenges
  - Limited amount of time for evaluations
  - Accuracy-time tradeoff in the evaluation
  - Fake parameters that increase the search space
  - No prior knowledge
  - Car can get damages



Front Wing 1 – 20: 10

Brake Balance (Rear : Front) 50% – 75%: 37.500 : 62.500

Rear Wing 1 – 20: 12

Gear Ratios

1st 28 : 64

2nd 35 : 64

3rd 42 : 64

4th 48 : 64

5th 54 : 64

6th 59 : 64

Pit Stop Strategy

None □

1 Stop □

2 Stops □

3 Stops ■

Stop on lap

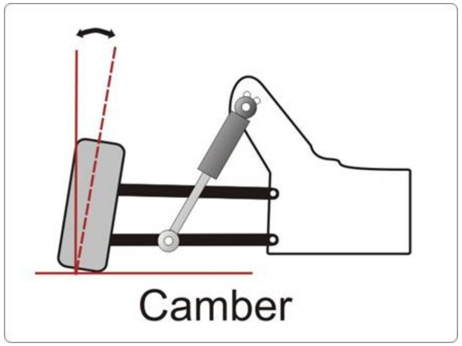
25%(3/9)

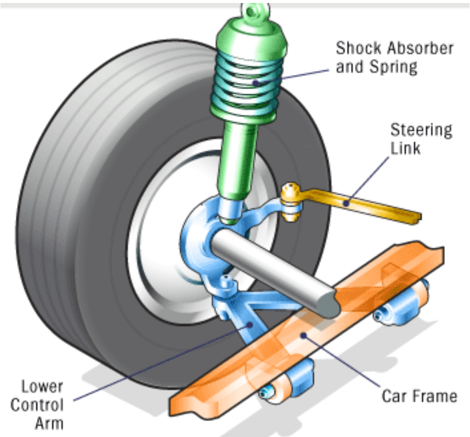
50%(4/9)

75%(7/9)

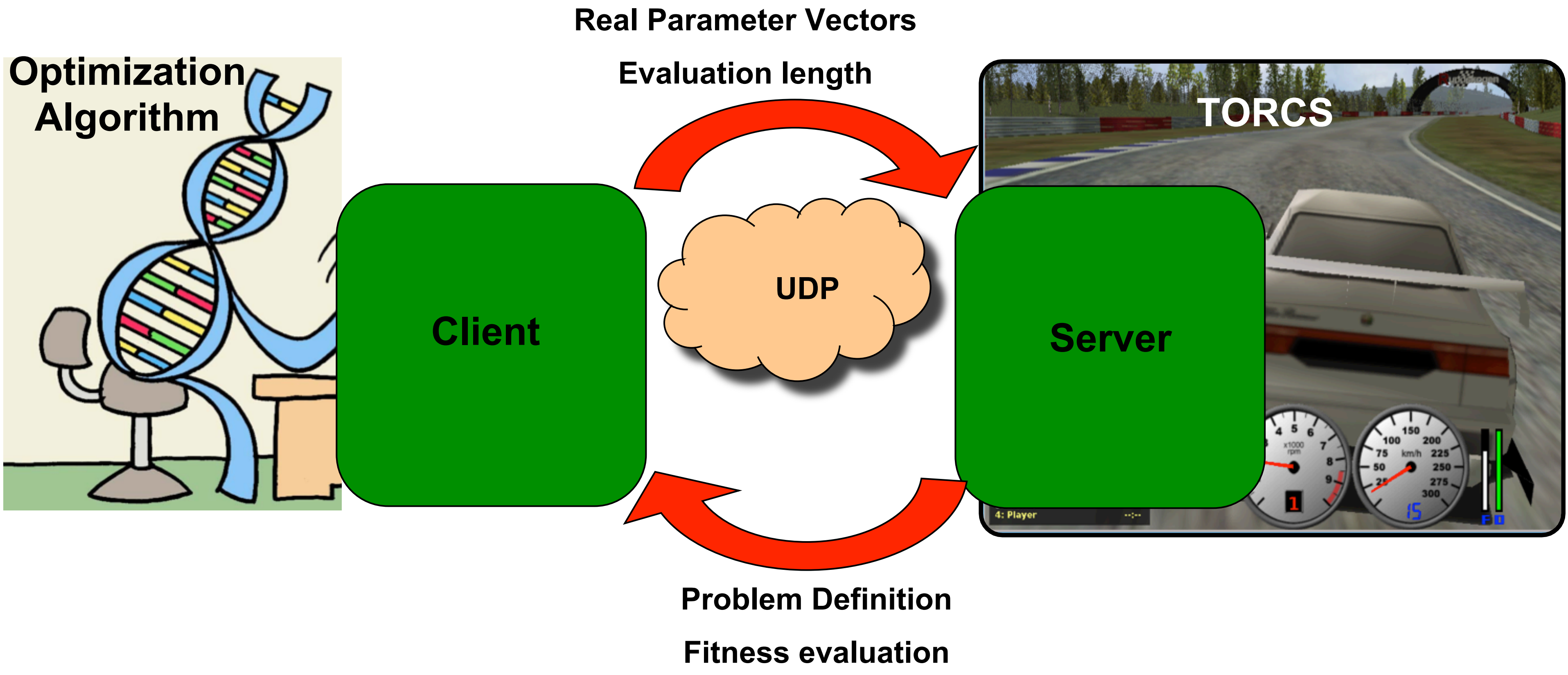
# Car Setup Optimization: Which parameters?

- A car presents many parameters that can be optimized:
  - Gear ratios
  - Rear/Front wing angle
  - Brakes
  - Rear differential
  - Rear/Front anti-roll bars
  - Wheels
    - Ride
    - Toe
    - Camber
  - Suspensions
    - Spring
    - Bell crank
  - …





# Car Setup Optimization: Framework



Optimization Algorithm

Real Parameter Vectors

Evaluation length

TORCS

Client

UDP

Server

Problem Definition

Fitness evaluation

# Car Setup Optimization: Results

- Organized at GECCO 2009 and at Evo* 2010
  - In 2009 won by Versari et al. (PSO)
  - In 2010 won by Munoz et al. (MOEA)

http://vimeo.com/10870222

# Physical Travelling Salesman Problem

# Physical Travelling Salesman Problem

- Extends the well-known Travelling Salesman Problem
- Add a physical dynamics to the movements of the salesman
- Solution consists of a long sequence of force vectors
- Run for the first time at GECCO 2005 and now WCCI 2012 and at CIG 2012
  - So far best entries are based on MCTS and A*

http://youtu.be/xV4DapXNgPE

# Games as testbed: a closing balance

- Game-based testbeds became very popular for several reasons:
  - challenging
  - entertaining
  - benchmarks not *just* game
- On the other hand…
  - often just a gamification of benchmarks
  - not easy to transfer obtained knowledge on a specific testbed
  - the need to defend games research is shrinking
- Trends
  - testbed more relevant for the game research (e.g., believability)
  - add humans in the loop

# Developing better games

## Why EC can improve games?

Improve the *poor* AI in games

Reduce the development cost/time

Allow knowledge-free AI development

# Why EC can't improve games?

~~Improve the *poor* AI in games~~

Nowadays AI is often very good

~~Reduce the development cost/time~~

AI development is not the most expensive task

~~Allow knowledge-free AI development~~

A black-box AI design means often boring games

# Evolutionary Design of NPC

- ❑ Early works in the field focused on beating the game…
- ❑ … now focus is more on non-player characters (NPC), i.e., characters not controlled by the player (either opponents or an allies)
- ❑ Design choices
  - How to represent the NPC?
  - How compute fitness?
  - Which evolutionary techniques?
- ❑ Some examples
  - Evolving Quake III bot
  - Evolving Racing Lines in Games

# Evolving bots for Quake III

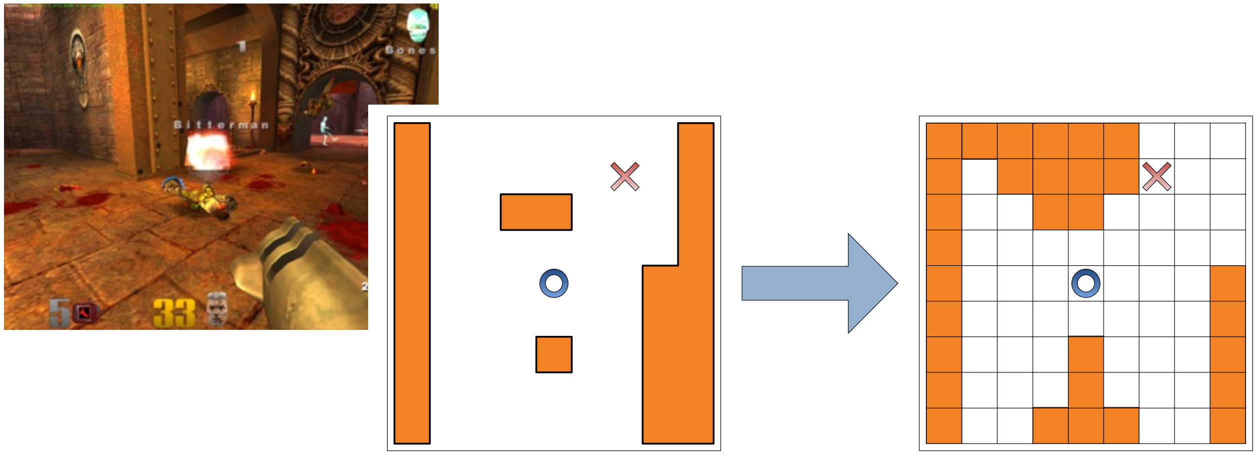
# Evolving bots for Quake III

Evolution of Reactive Rules in Multi-Player Computer Games Based on Imitation, Priesterjahn et al., 2005.

http://youtu.be/mKdIi9BM_RI

# Evolving bots: representation

- How to represent the game environment?
  - Collect information with raycasting
  - Discretize local area around the NPC

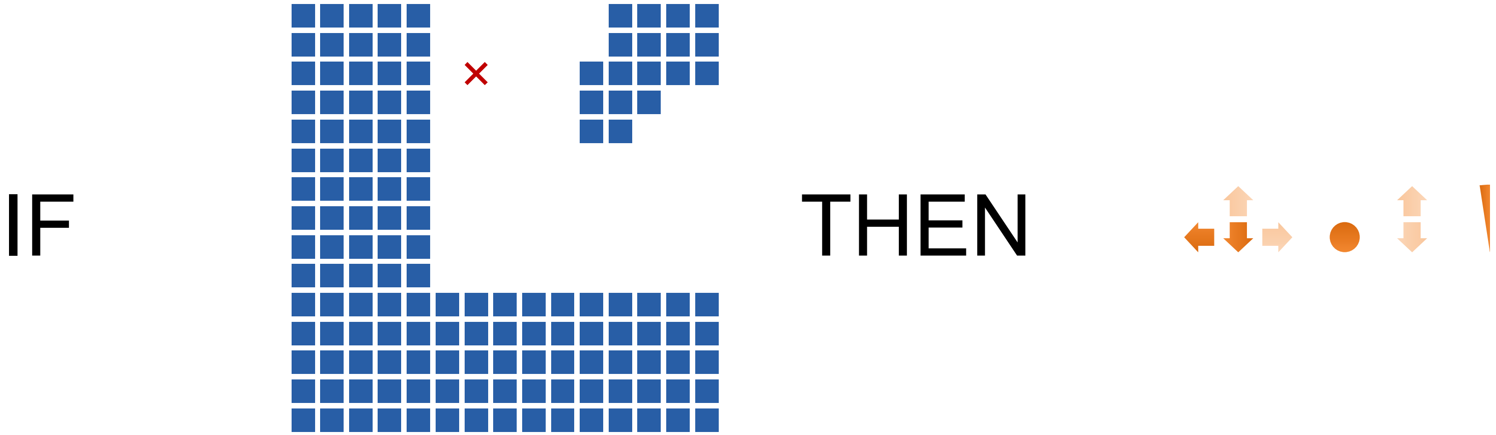


Game

Discrete Representation

# Evolving bots: representation (2)

- How to represent an NPC strategy?
  - population of if-then rules
  - game environment is matched against the rules
  - rule with the closest matching is applied



IF

THEN

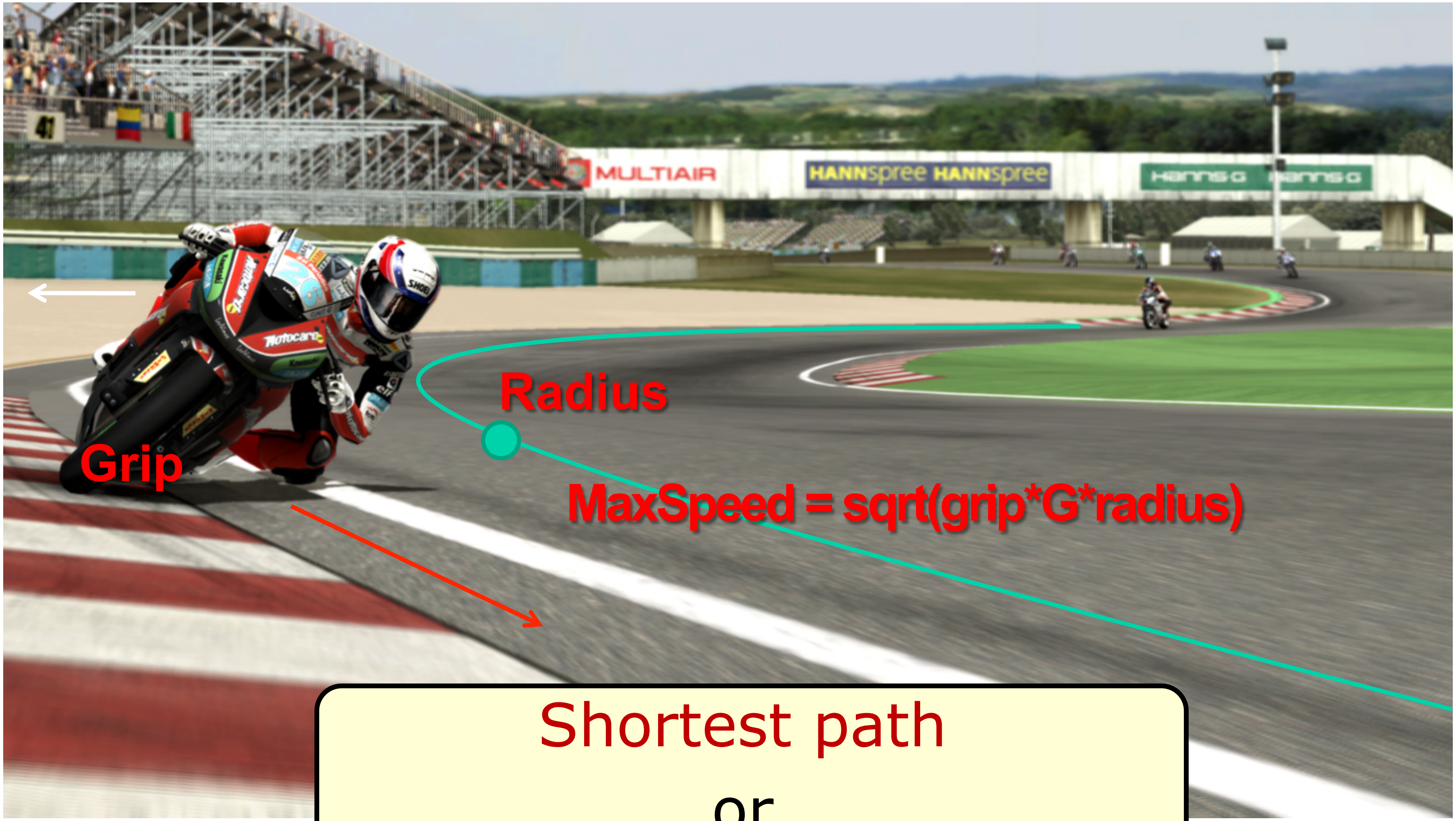
# Evolving bots: approach

- How to find the best rules?
  - Real-players data used to build a rule-base
  - Individuals are generated by selecting a random set of rules from the rule-base
  - GA is applied to evolve the best set of rules
  - Recombination works on the sets of rules
  - Mutation works on the single rules
  - Fitness is computed as

$$\text{fitness} = \text{damage dealt} - \text{damage received}$$

# Evolving Racing Lines for Racing Games

# Racing Liens: the problem



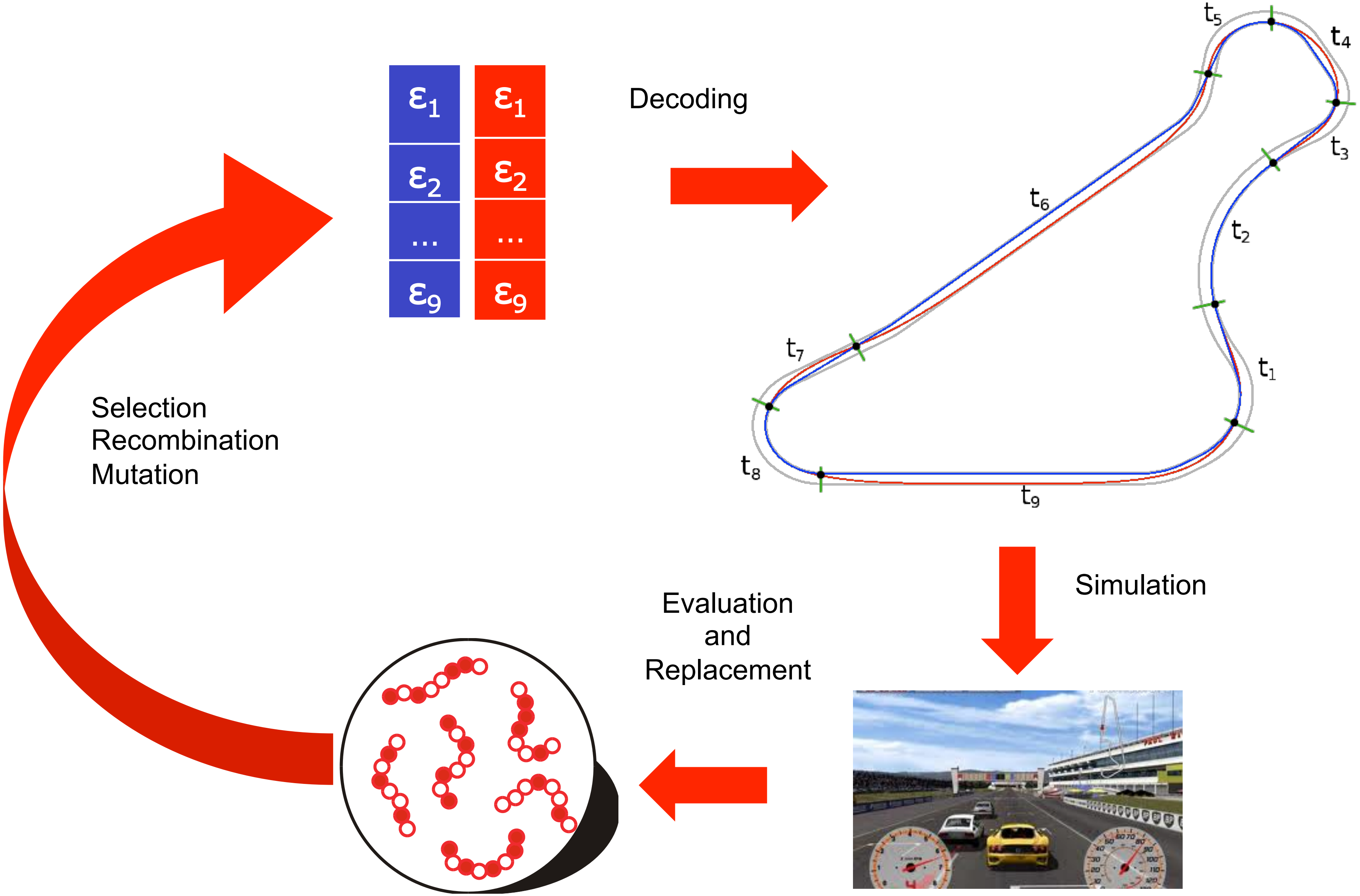
Shortest path

or

minimum curvature ?

# Racing Lines: standard approach

# Racing Lines: evolutionary approach



Decoding

Simulation

Evaluation
and
Replacement

Selection
Recombination
Mutation

What do we learn from the literature about evolutionary design of NPC?

# Evolutionary design of NPC: Representation

- ❑ Parameterized strategy
  - ► requires strong domain knowledge
  - ► prevents emergent behaviors
  - ► easy to optimize and reliable
- ❑ Rules or trees
  - ► requires discrete actions or well defined basic behaviors
  - ► allows to integrate existing knowledge
  - ► allows some emergent behaviors
- ❑ Decision function (e.g., NN)
  - ► very few domain knowledge required
  - ► difficult to integrate existing knowledge
  - ► definitely allows emergent behaviors
  - ► might lead to unreliable results

# Evolutionary design of NPC: Fitness function and technique

- Fitness function
  - generally based on in-game statistics
  - cost/significance trade-off
  - often noisy or non-deterministic
- Evolutionary technique depends on the representation used
  - Parameterized strategy → ES, GA, PSO, etc.
  - Rules or trees → LCS, GP, EP, etc.
  - Decision function → Neuroevolution

# Evolutionary design of NPC: Expect the unexpected

- EvoStar 2011: Mr Racer bot (Quadflieg et al.) good but suffers from default clutch control
  - First approach similar to winner’s clutch control: speed based
  - Autopia (winner) closes clutch below 70 km/h
  - We adapt closing (logistic) function with a bit more freedom
  - Result: using the clutch until 180 km/h is profitable
  - We would be much worse with restriction to 70 km/h

http://youtu.be/Kk1mC6mZjVc

Besides NPCs…

# Besides evolutionary design of NPCs

- Several applications of CI to games
  - believability
  - adaptivity and in-game learning
  - analysis of player behaviors
  - improving game components
- In particular, notable applications of EC includes
  - evolving believable NPCs with MOEA (Togelius et al.)
  - real-time neuroevolution in games (NERO, Stanley et al.)
  - neuro-evolutionary preference learning (Yannakakis et al.)
  - automatic camera control (Burelli et al.)

# Developing *innovative* games

# Evolving Game Content

- Challenges:
  - How to represent the content?
  - Which is the best representation to be evolved with genetic algorithms?
  - How to evaluate the game content?

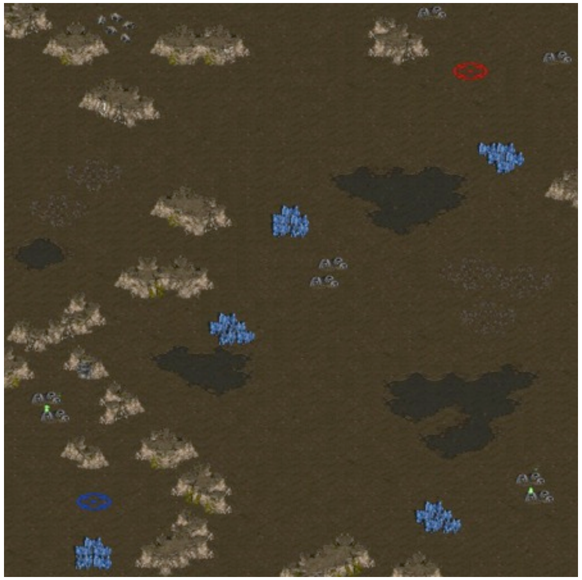
- Case studies:
  - Evolving Starcraft and Maps
  - Evolving tracks for racing games
  - Evolving maps for First Person Shooters

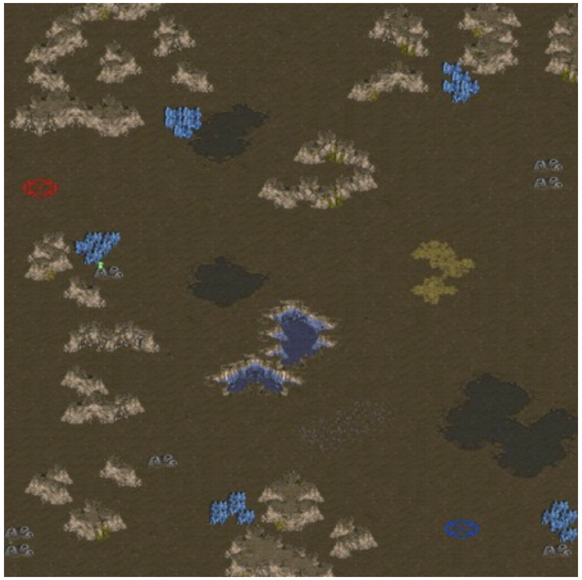
# Multi-objective optimization in games

- Great potential (→ detect tradeoffs), few uses:
  - Schrum/Miikkulainen 2008: constructing complex NPC behavior
  - Agapitos 2008: generating diverse opponents
  - Togelius et al. 2010: exploration of StarCraft map space
  - Bin Tan/Theo/Anthony 2010: evolution of neural Go players
  - Preuss/Quadflieg/Rudolph 2011: multi-objective track selection
- We may have missed some, but still…

# Multi-objective tradeoff exploration

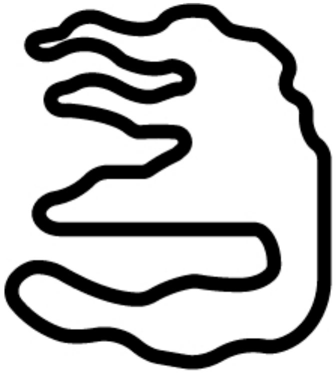
- Starcraft Maps (Togelius et al., 2010)
  - 8 objectives: base location, ressource fairness, choke points etc.
  - Unclear which objectives make sense
  - But single objectives can be discussed with users
  - We enforce formalization
  - Innovation: users may be wrong (e.g. fair and asymmetric maps)
  - Exploration via multi-objective optimization: conflicts, tradeoffs
  - This example: PCG, other uses similar

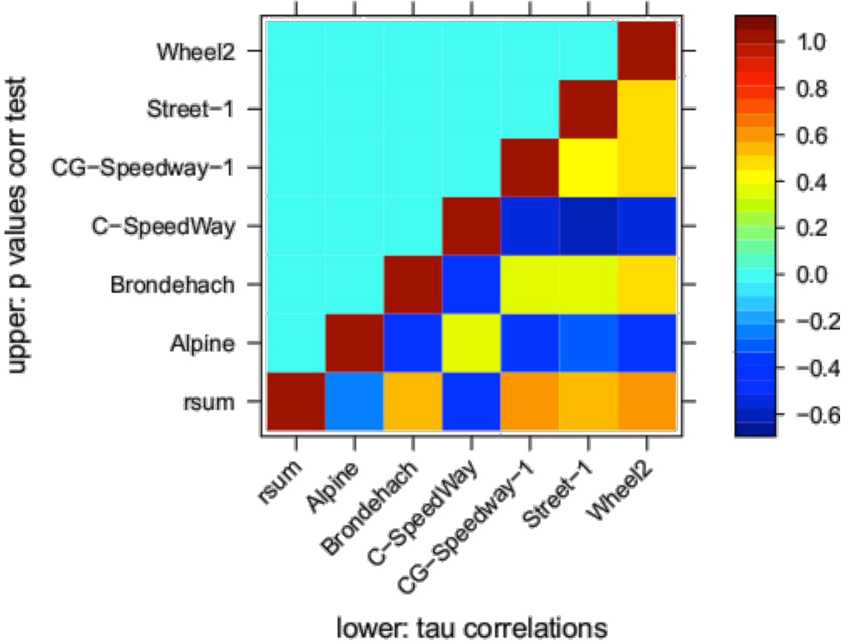




# Multi-objective representative selection

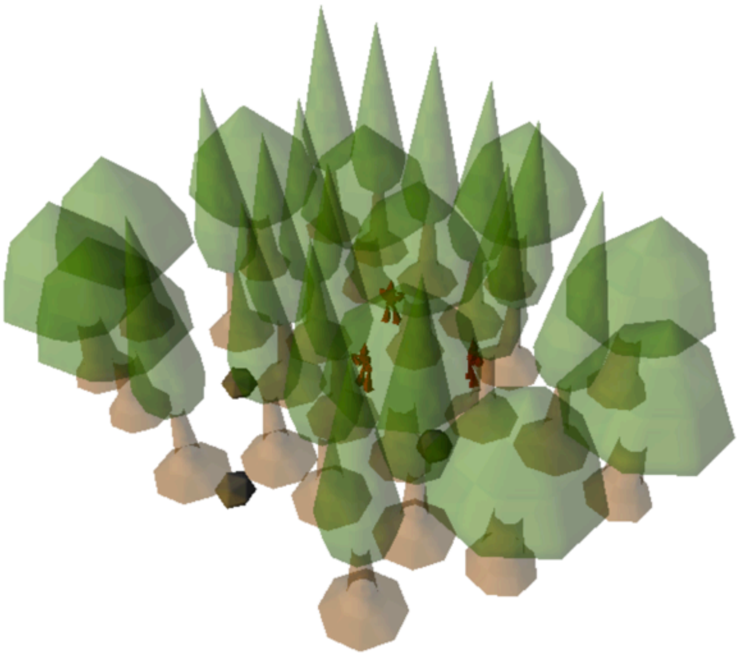
- Small number of driver configurations that add up well for different tracks (Preuss/Quadflieg/Rudolph, 2011)
  - Full MO run with 6 objectives very expensive (runtime, instability)
  - Single-objective on 2  tracks: very different solutions
  - We evaluate the best solutions of both on all tracks
  - Correlation analysis (rank based): similarity of tracks
  - Simple and working, but much to do here…

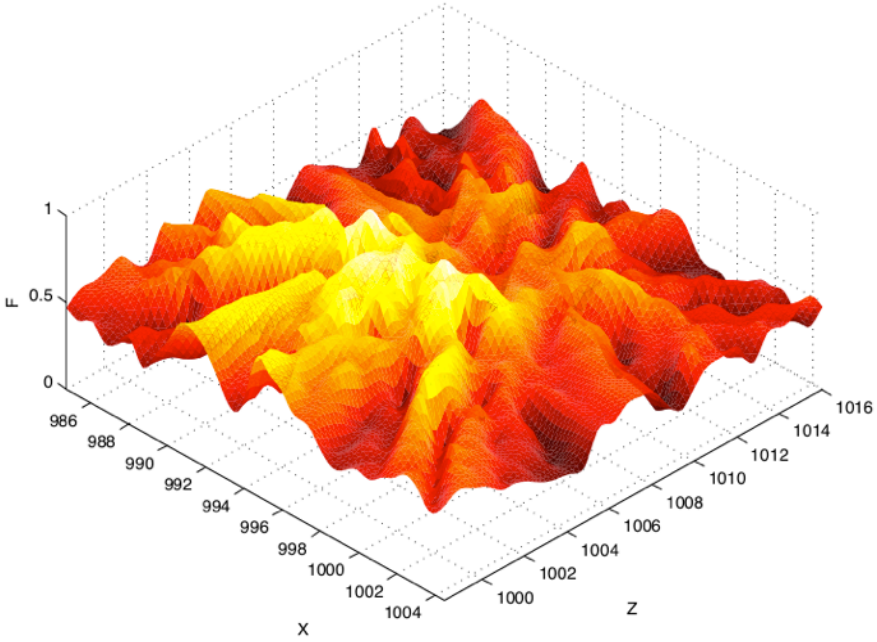




# Many objectives: camera positioning

- At first: weighted single-objective (Preuss/Burelli/Yannakakis, 2012)
  - Vision: realtime multi-objective (often 3-9), diverse solutions
  - Currently not possible (time), but learned a lot about problem
  - Very interesting benchmark for optimization methods

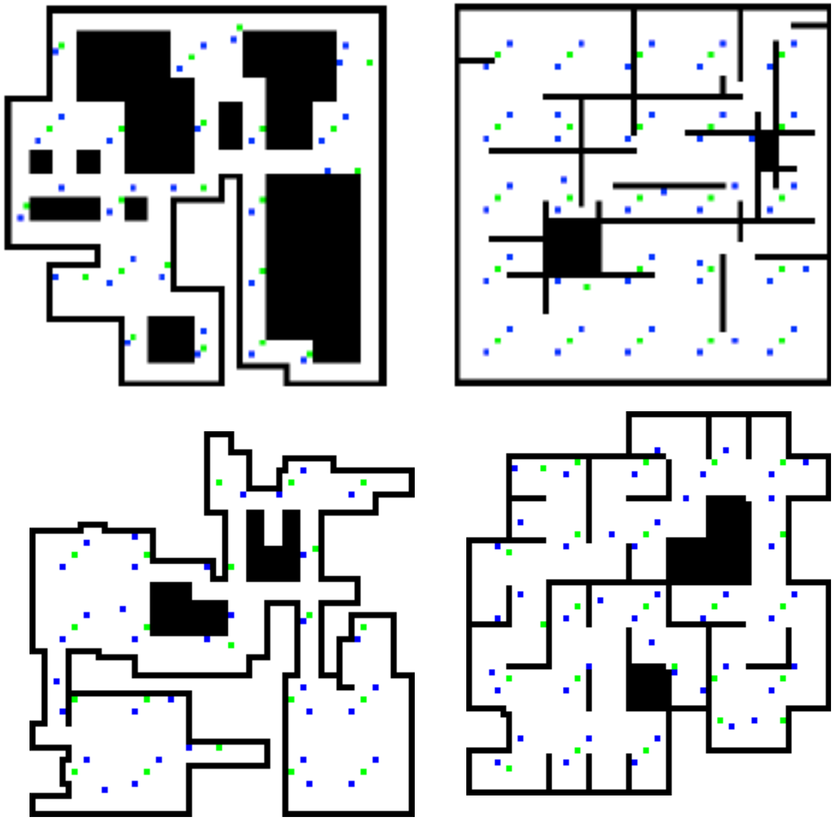


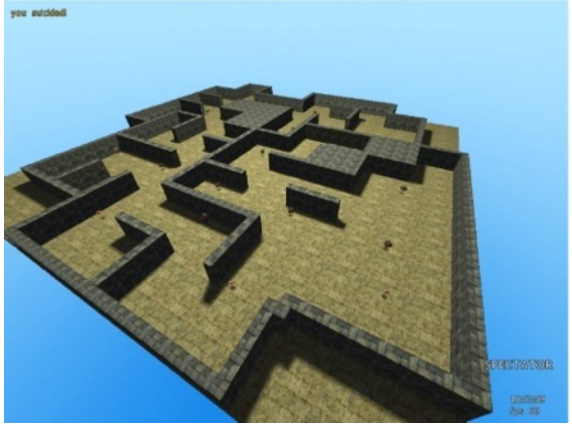


## Evolving maps for FPS

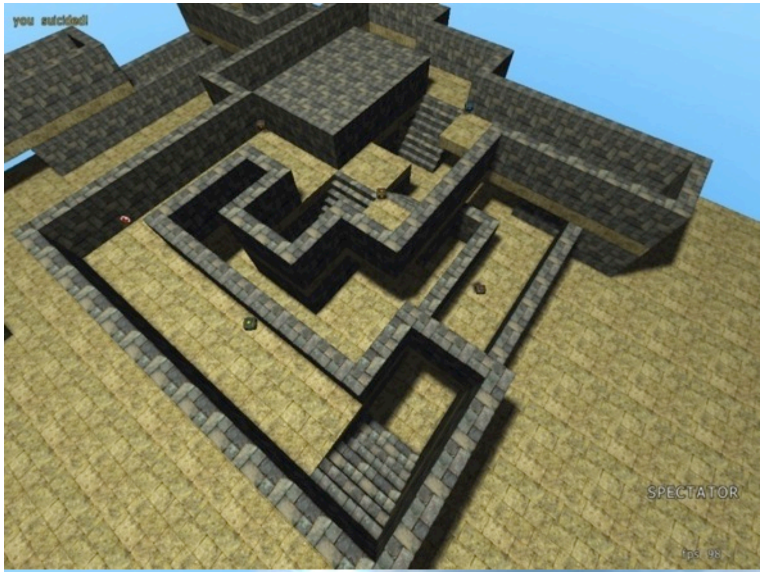
# Evolving FPS Maps: overview

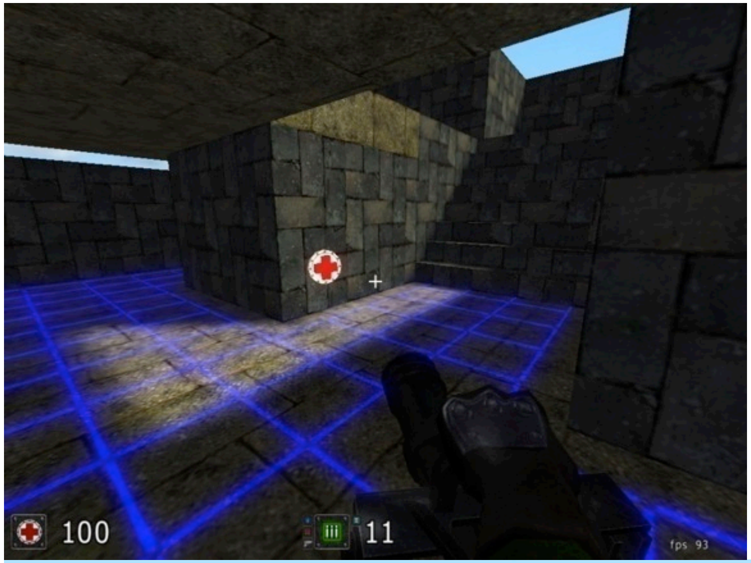
- ❑ Cardamone et al. evolved maps for CUBE, an open source First Person Shooter
- ❑ Four different representations were compared
- ❑ Fitness based on game statistics computed using NPCs

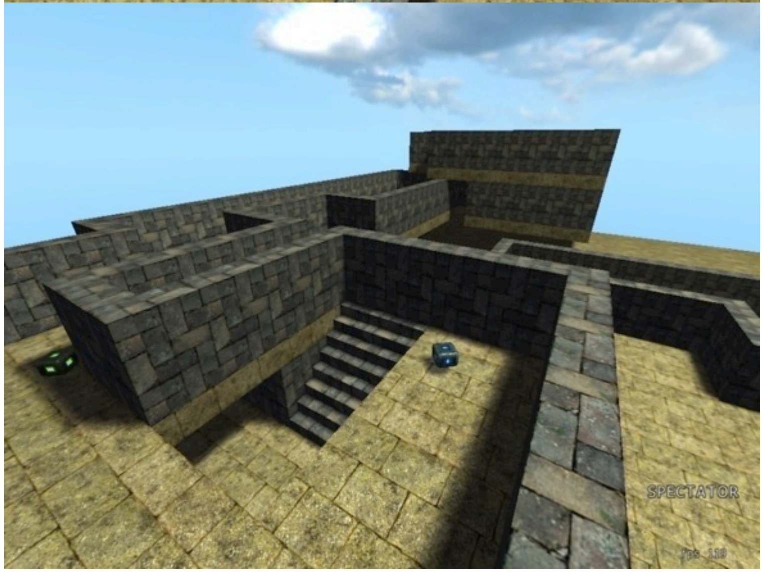


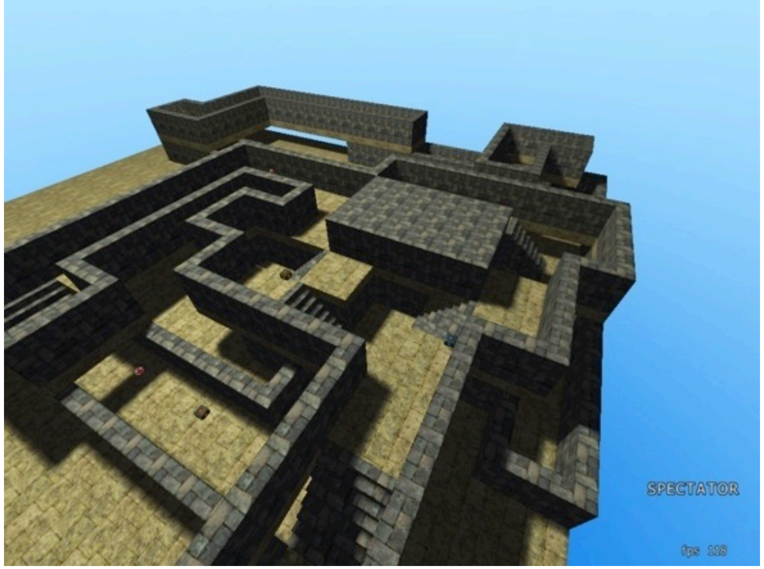


# Evolving FPS Maps: examples





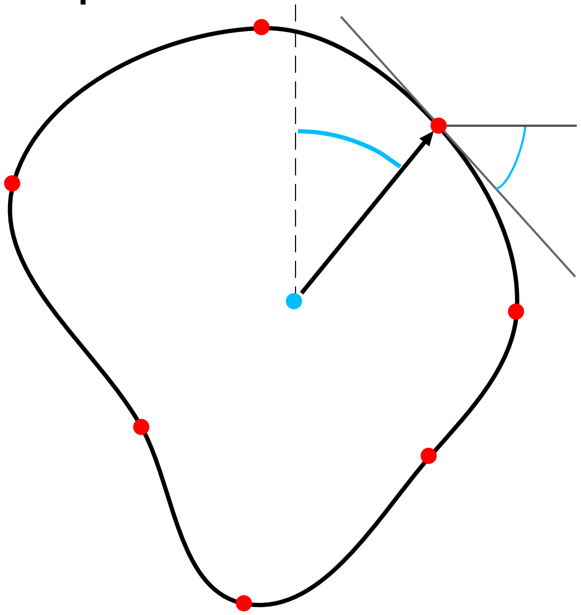




# Evolving Racing Tracks

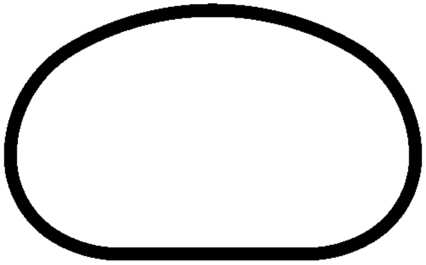
# Evolving Racing Tracks: the approach

- ❑ Apply a genetic algorithm to evolve the shape of the tracks for a realistic racing game

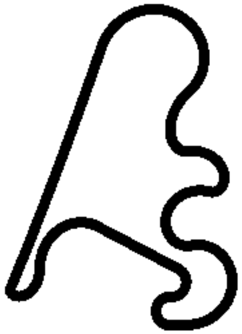


- ❑ Representation based on a list of way points in polar coordinates

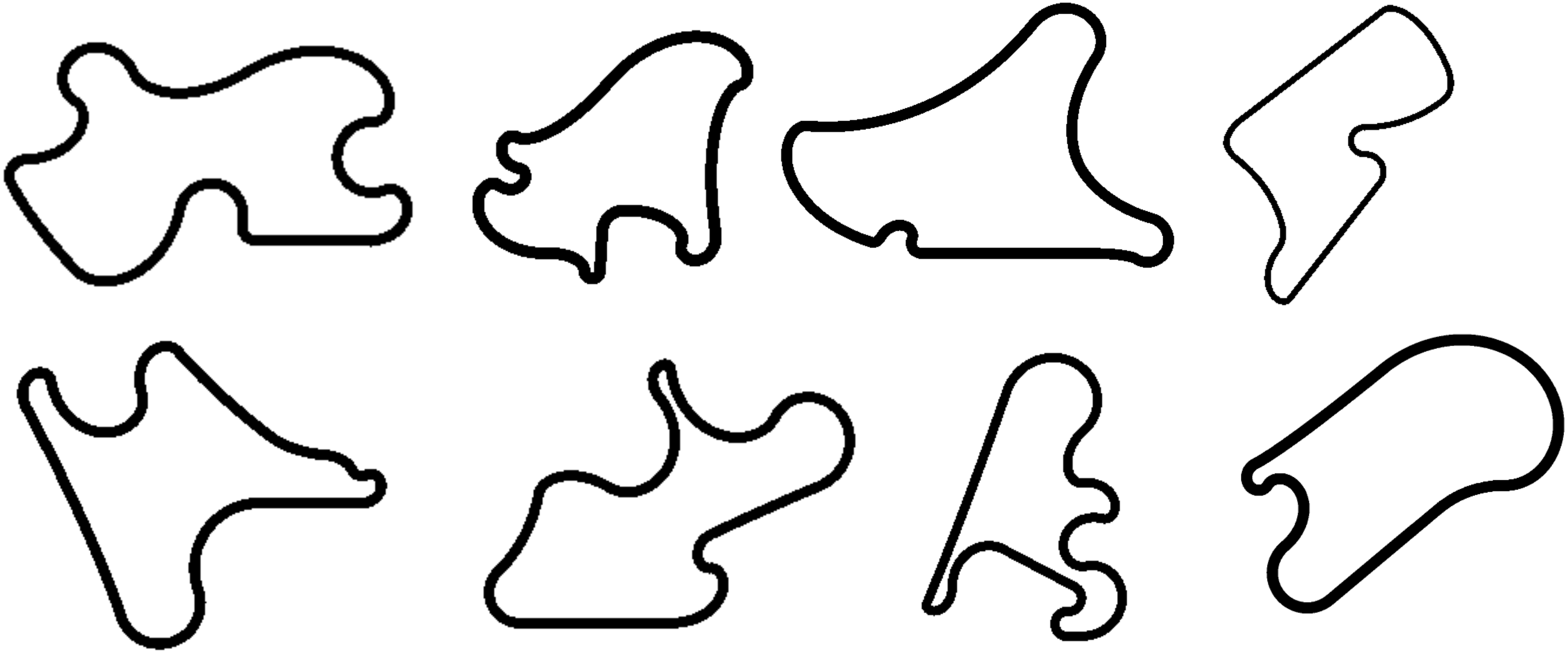
- ❑ Fitness based on diversity:
  - Entropy of speeds
  - Entropy of curvatures
  - Both



VS



# Evolving Tracks: examples



How to put users in the loop?

A couple of examples…

# Galactic Arms Race (GAR)

http://gar.eecs.ucf.edu/

http://eplex.cs.ucf.edu/movies/gar_promo2.wmv

# TORCS/Speed Dreams Tracks Generator

http://trackgen.pierlucalanzi.net

http://youtu.be/YFOa7L3oBwM

http://youtu.be/0_W4jHN2h2Q

http://youtu.be/Si_u_43HJnM

http://youtu.be/3AzjMtRDnBo

# Software Platforms

## Racing Games

Point-to-point

RARS

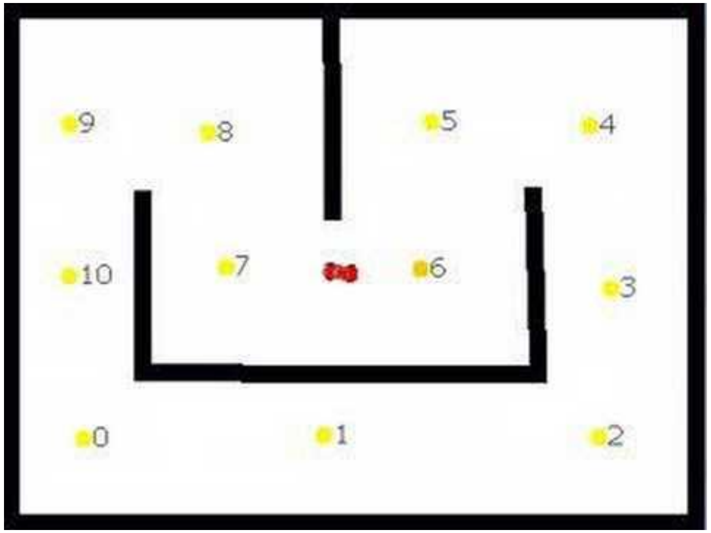
TORCS / Speed Dreams

VDrift

Simulated Car Racing

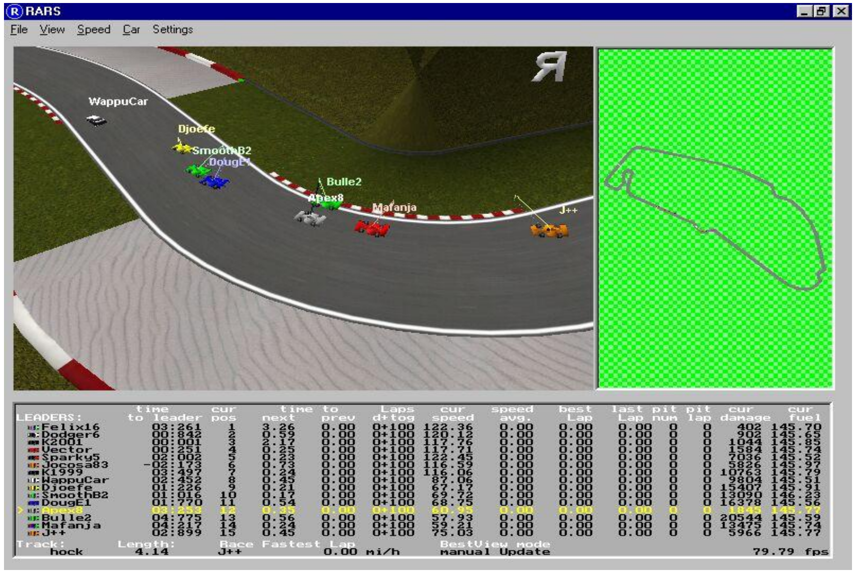
# Point-to-point car racing and RARS

## Point-to-point



- ❑ Simple Java racing environment (Togelius et al., 2005)
- ❑ Used for competitions at CIG 2007 and at CEC 2007
- ❑ http://julian.togelius.com/cec2007competition/

## RARS



```
                   time       cur    time  to     Laps    cur     speed    best    last pit pit   cur     cur
LEADERS:        to leader     pos    next   prev  d+tog   speed    avg.     Lap     Lap  num lap damage   fuel
   Felix16        03:261       1     3.26   0.00  0+100  122.36    0.00    0.00    0.00   0   0    402  145.70
   Dodger6        00:842       2     0.59   0.00  0+100  120.12    0.00    0.00    0.00   0   0    902  145.65
   K2001          00:001       3     2.17   0.00  0+100  117.76    0.00    0.00    0.00   0   0   1044  145.85
   Vector         00:251       4     0.25   0.00  0+100  117.71    0.00    0.00    0.00   0   0   1584  145.74
   Sparky5        02:002       5     0.23   0.00  0+100  122.45    0.00    0.00    0.00   0   0   7036  145.52
   Jocosa83      -02:173       6     0.73   0.00  0+100  116.59    0.00    0.00    0.00   0   0   5826  145.97
   K1999          03:497       7     0.24   0.00  0+100  116.06    0.00    0.00    0.00   0   0  10763  145.79
   WappuCar       02:452       8     0.45   0.00  0+100   87.06    0.00    0.00    0.00   0   0   9804  145.51
   Djoefe         01:226       9     0.21   0.00  0+100   67.17    0.00    0.00    0.00   0   0  15407  145.91
   SmoothB2       01:016      10     0.17   0.00  0+100   69.72    0.00    0.00    0.00   0   0  13090  146.23
   DougE1         01:770      11     0.54   0.00  0+100   68.75    0.00    0.00    0.00   0   0  16378  145.56
 > Apex8          03:253      12     0.35   0.00  0+100   60.95    0.00    0.00    0.00   0   0   1845  145.77
   Bulle2         04:775      13     0.56   0.00  0+100   57.93    0.00    0.00    0.00   0   0  20544  145.59
   Mafanja        04:217      14     0.24   0.00  0+100   59.21    0.00    0.00    0.00   0   0  13475  145.74
   J++            02:899      15     0.45   0.00  0+100   75.03    0.00    0.00    0.00   0   0   5966  145.77
Track:        Length:        Race Fastest Lap            BestView mode
     hock         4.14         J++         0.00 mi/h     manual Update                          79.79 fps
```

- ❑ Open source 3D racing simulator
- ❑ Designed to enabled pre-programmed AI drivers to race against
- ❑ http://rars.sourceforge.net/

# TORCS / Speed Dreams and VDrift

## TORCS / Speed Dreams

Laps: 1 / 30
Best: 00:00 +00:00
Time: 34:50
<- Inferno 6 +00:00
-> berniw two 7 +00:00

1: berniw 4 00:00
2: berniw 7 --:--
3: Tita --:--
4: berniw 1 --:--
5: bt 7 --:--
6: LLiaw --:--
7: Inferno 7 --:--
8: Inferno 2 --:--
9: Inferno 4 --:--
16: Olethros 6 --:--

- ❑ Accurate physics engine specifically developed for racing (traction, damage, aerodynamics,…)
- ❑ Wide community of users providing tracks and other game content
- ❑ [http://torcs.sourceforge.net](http://torcs.sourceforge.net)

## VDrift

Lap time: 00:51.322
Last lap: --:--.---
Best lap: --:--.---
Lap 1/1
Place 1/10

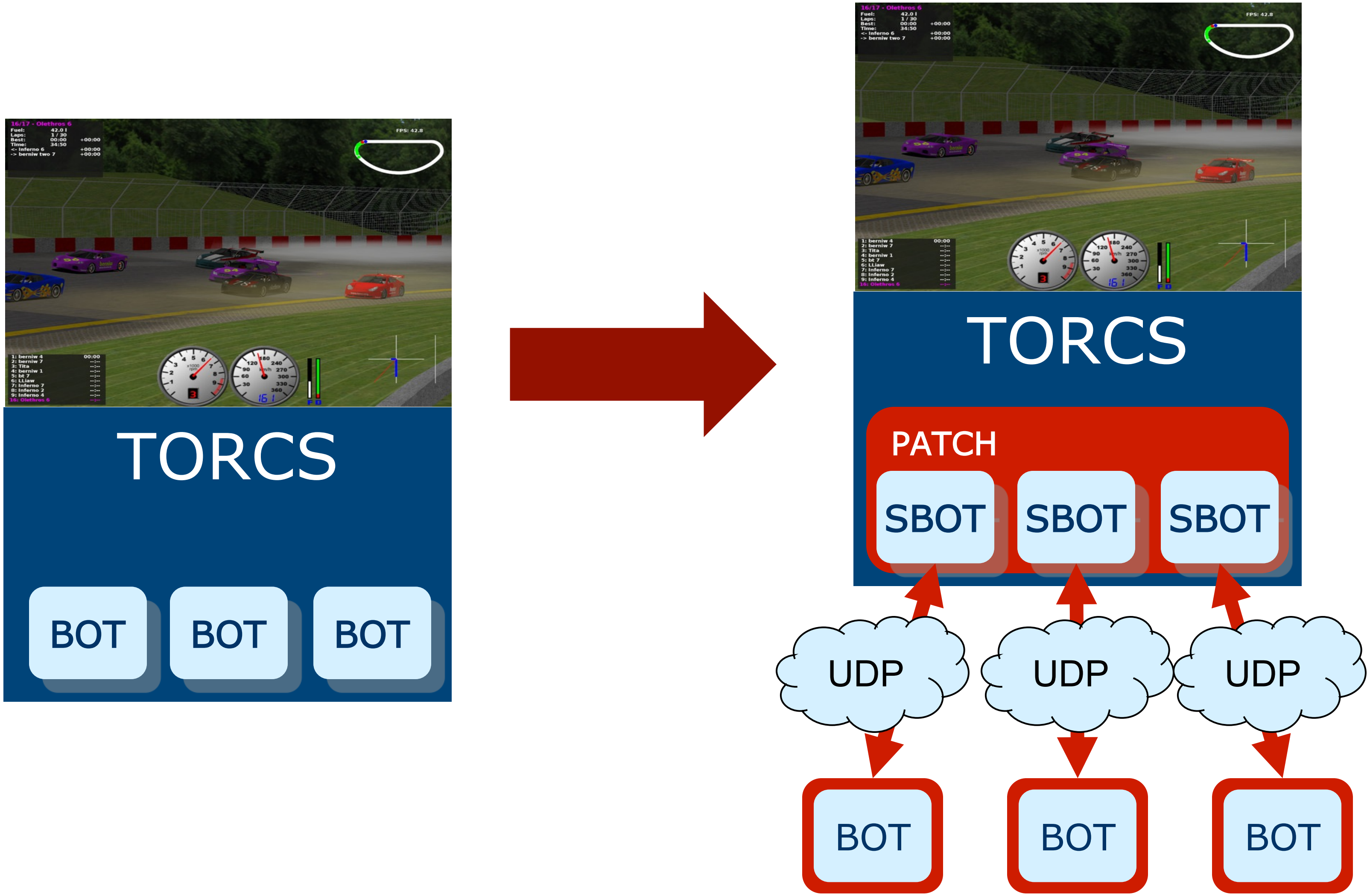
- ❑ Use Bullet, an open source physics engine featuring 3D collision detection, soft and rigid body dynamics
- ❑ Accurate simulation of loss of traction (drift)
- ❑ [http://vdrift.net](http://vdrift.net)

# Simulated Car Racing

- Simulated Car Racing (SCR) requires the development of a driver for TORCS (hand-coded, learned, evolved, …)
- SCR typically involves 9 races organized in three different legs during three major conferences
- Teams are awarded based on their score in each conference competition
- At the end, the team with highest overall score wins the championship
- SCR has been organized since 2009

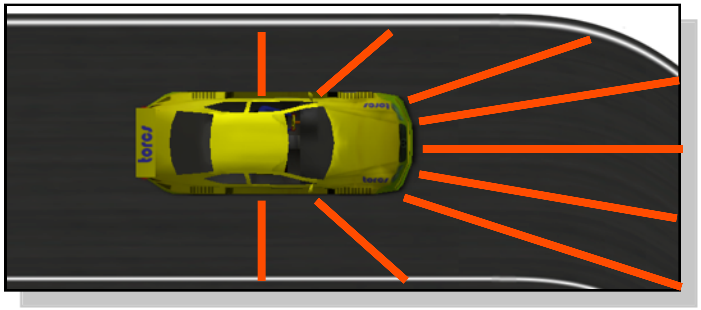
- http://games.ws.dei.polimi.it/competitions/scr/

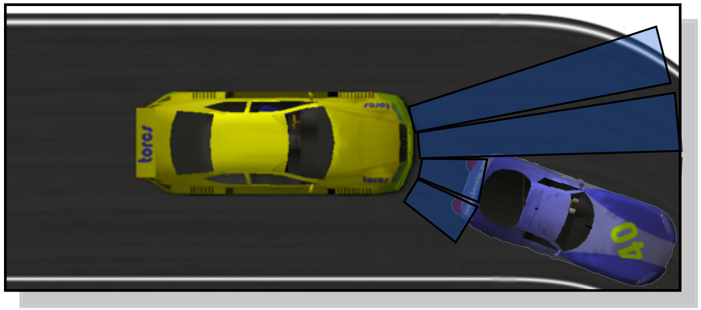
# Simulated Car Racing: architecture

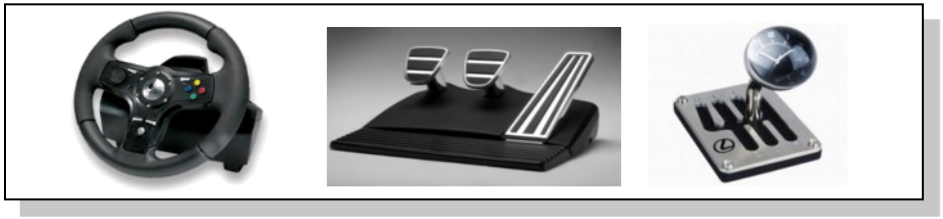


# Simulated Car Racing: sensors and actuators

- Rangefinders for edges on the track and opponents (with noise)
- Speed, RPM, fuel, damage, angle with track, distance race, position on track, etc.







- Six effectors: steering wheel [-1,+1], gas pedal [0, +1], brake pedal [0,+1], gearbox {-1,0,1,2,3,4,5,6}, clutch [0,+1], focus direction

## First Person Shooters

ioquake3

Cube

Unreal Tournament

2K Bot Prize

# ioquake3 and Cube

## ioquake3

- ❑ Bug-free and enhanced implementation of the id Software’s Quake 3 engine
- ❑ Used in several game projects as well as in several academic projects
- ❑ http://ioquake3.org

## Cube

- ❑ Cube 2: Sauerbraten is a free multiplayer/singleplayer first person shooter
- ❑ Allow map/geometry editing to be done dynamically in-game
- ❑ http://sauerbraten.org/

# Unreal Tournament

- ❑ Very popular series of multiplayer FPS by Epic Games
- ❑ Does not require expensive hardware to run
- ❑ Can be easily customized with scripting
- ❑ http://www.unrealtournament.com/
- ❑ Unreal Wiki: http://wiki.beyondunreal.com/

# 2K BotPrize

- The BotPrize competition challenges programmers/ researchers/hobbyists to create a bot for UT2004 (a first-person shooter) that can fool opponents into thinking it is another human player.
- The competition organized by P. Hingston has been sponsored by 2K games since 2008, and the $5000 major prize is yet to be claimed.
- [http://www.botprize.org/](http://www.botprize.org/)

## Strategy Games

Starcraft and Starcraft Competition
Stargus and Wargus

# Starcraft Competition

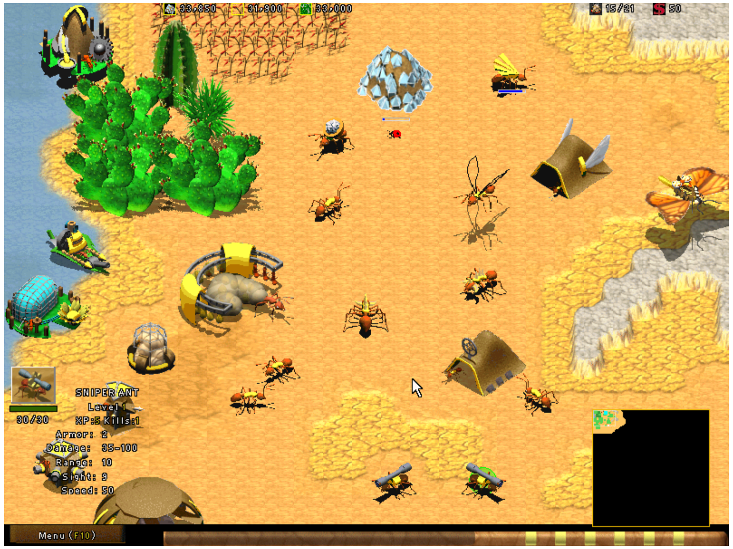
- Held at AIIDE and CIG conferences since 2010, setup differs slightly: AIIDE maps are known beforehand, CIG maps
- Bots attached to Starcraft via 3rd person hack BWAPI: http://code.google.com/p/bwapi/
- Active scene of around 20 bot developers/teams
- Both competitions won by Skynet bot in 2011
- Current limitations: most bots are not very adaptive to opponent strategy

http://youtu.be/xXsx1ma3_ko

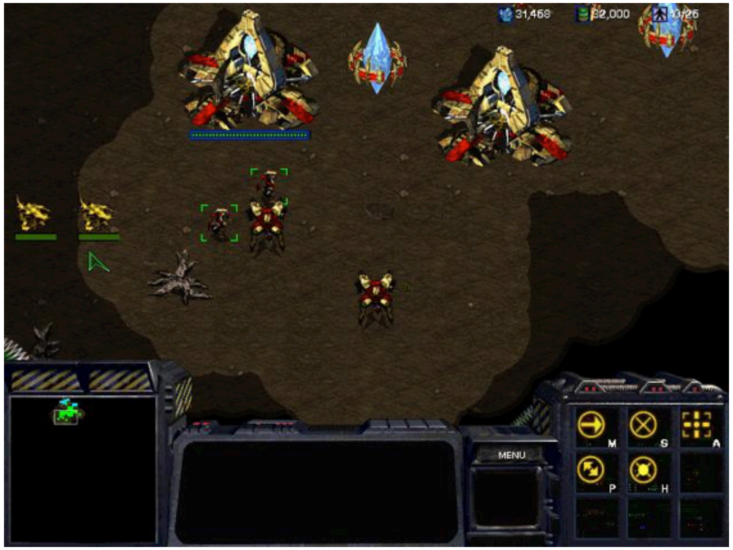
http://webdocs.cs.ualberta.ca/~cdavid/starcraftaicomp
http://ls11-www.cs.uni-dortmund.de/rts-competition/starcraft-cig2012

# Stratagus, Stargus and Wargus

- **Stratagus** is a free cross-platform real-time strategy gaming engine.
- It includes support for playing over the internet/LAN, or playing a computer opponent.
- It is easily configurable and can be used to create games with a wide-range of features specific to your needs.
- Stargus and Wargus are mods that allow to play the popular **Starcraft** and **Warcraft** games with Stratugus engine
- http://stratagus.com/
- http://wargus.sourceforge.net/
- http://stargus.sourceforge.net/



SNIPER ANT
Level 1
XP:5 Kills:1
Armor: 2
Damage: 35-100
Range: 10
Sight: 9
Speed: 50



## Action Games

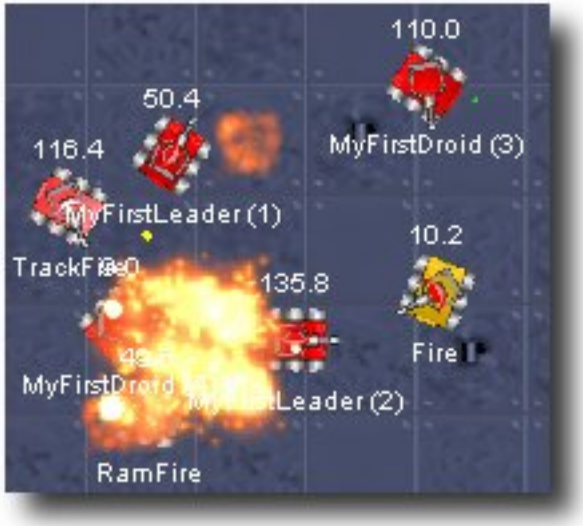
Robocode

Infinite Mario

Ms. Pac-Man

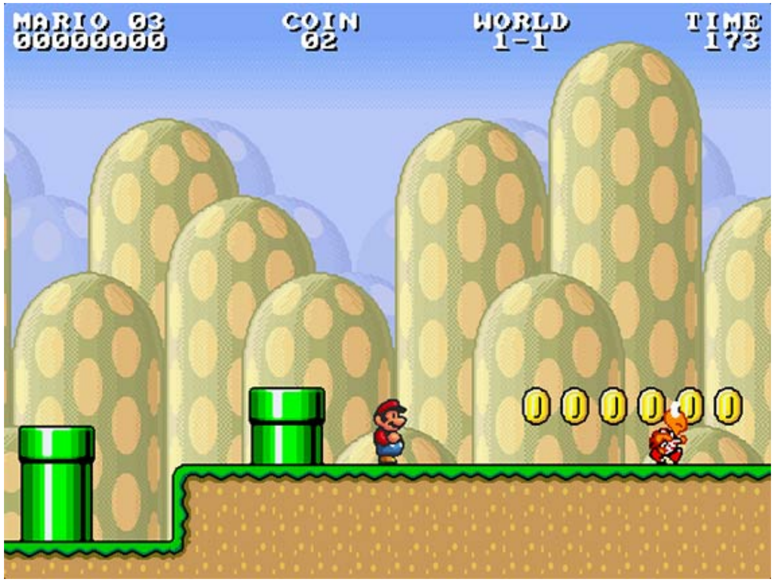
# Robocode

- ❑ Robocode is a programming game, where the goal is to develop a robot battle tank to battle against other tanks
- ❑ The robot tanks can be developed either in Java or .NET.
- ❑ Battles can be either run in real-time and displayed on the screen or run in a batch mode without visualization.
- ❑ It has a large community and features an on-line tournament system to rank developed tanks
- ❑ Official page: http://robocode.sourceforge.net/
- ❑ Robo wiki: http://robowiki.net/



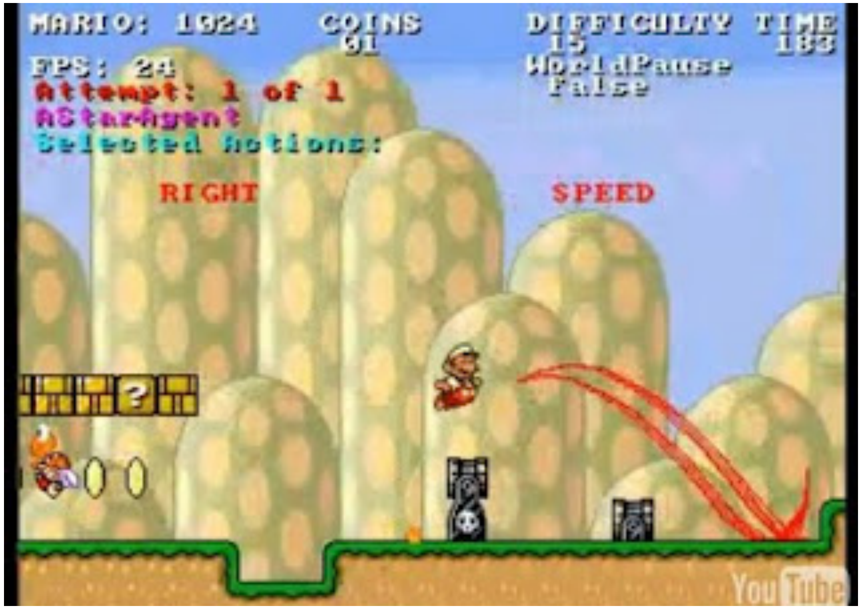
# Infinite Mario

- Infinite Mario Bros is a Java-based browser game developed by Markus Persson for Super Mario themed contest
- It provides unending 2D platforming action: all areas and level selection maps are generated from a random seed.
- The game and the entire source code is available for download at [http://www.mojang.com/notch/mario/](http://www.mojang.com/notch/mario/)



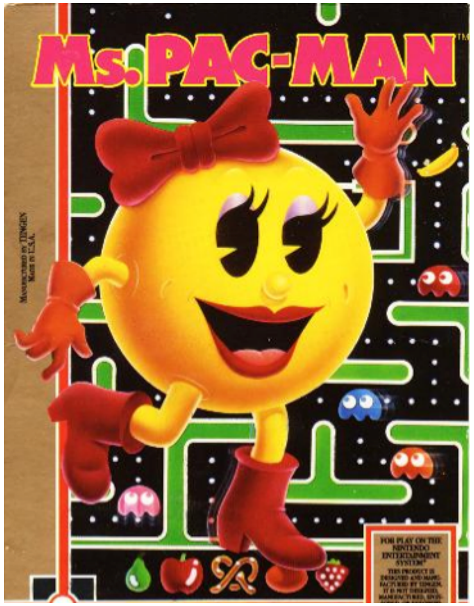
# Mario AI competition

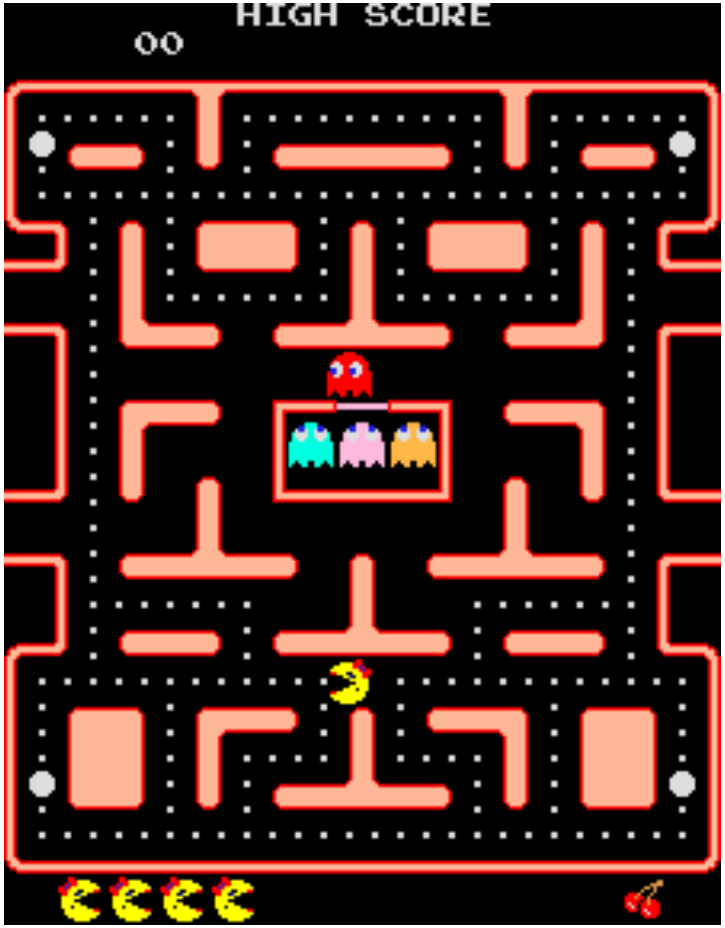
- ❑ Mario AI competition has been organized by Togelius et al since 2009 as part of the major conferences of the field
- ❑ It consists of four tracks with different goals:
  - ▸ Gameplay
  - ▸ Learning
  - ▸ Level Generation
  - ▸ Turing Test
- ❑ Competitors are provided with very effective Java APIs based on Mario Infinite



# Ms. Pac-Man

- A sequel of Pac-Man with a very similar gameplay
- It features four different mazes
- Involves non-deterministic opponents
- http://www.webpacman.com/





# Ms. Pac-Man competitions

- ❑ Ms. Pac-man vs Ghosts Competition (Rohlfshagen, Robles and Lucas)
  - allows you to develop AI controllers for either Ms Pac-Man or for the ghosts
  - based on a powerful Java framework developed by the organizers
  - http://www.pacman-vs-ghosts.net/
- ❑ Ms Pac-Man Screen Capture Competition (Lucas)
  - The aim of this competition is to provide the best software controller for the game of Ms Pac-Man.
  - It is based on the original version of the game
  - About 15 times per second the controller receives a pixel map of the Ms. Pac-Man window and it has to respond with the direction of the joystick.
  - http://cswww.essex.ac.uk/staff/sml/pacman/PacManContest.html

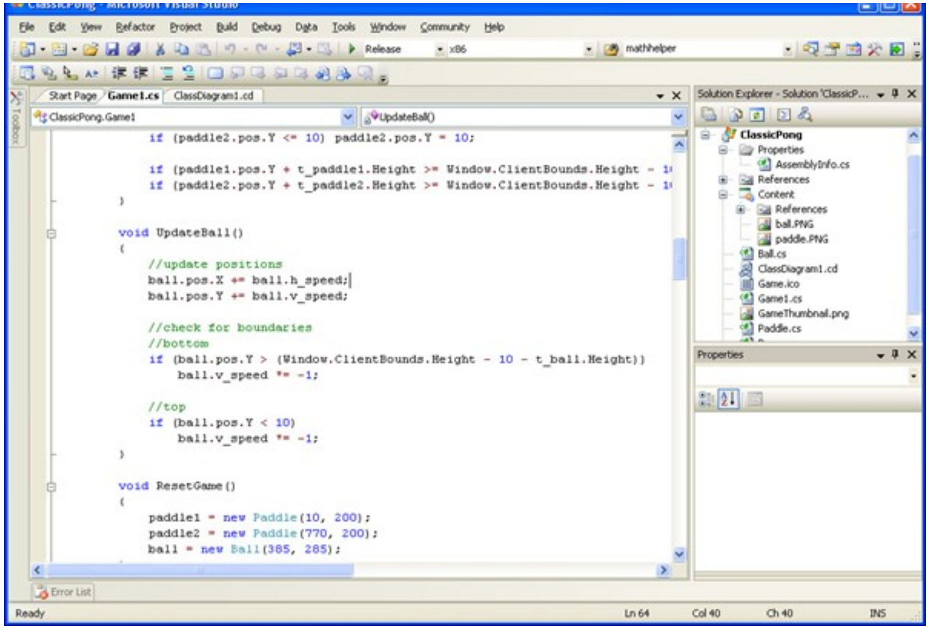
## Develop your own game framework

XNA

Unity

# XNA

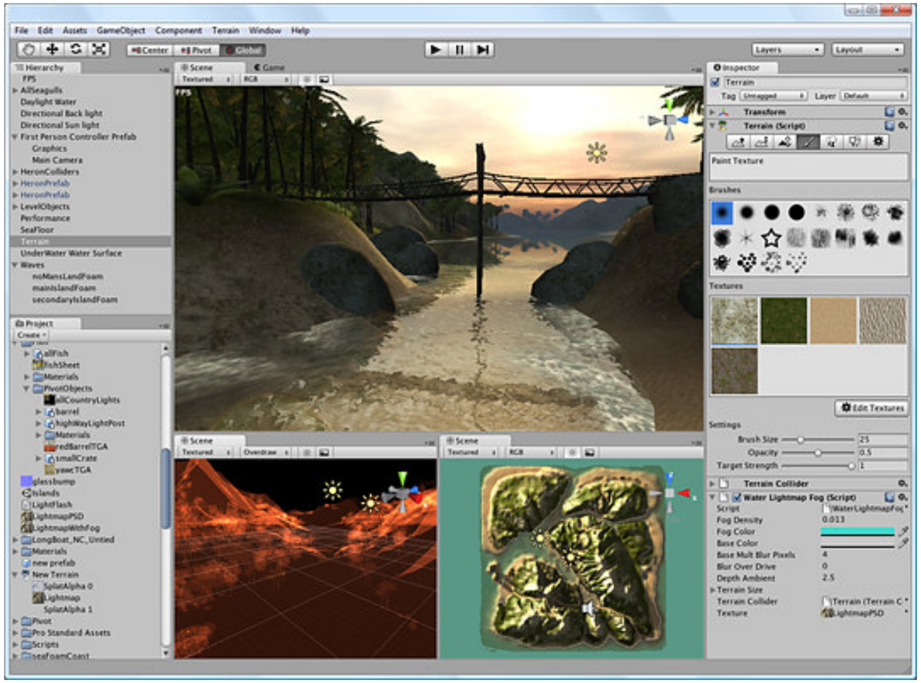
- ❑ Framework developed by Microsoft to simplify the development of games
- ❑ The development with XNA is performed in C# under .NET framework
- ❑ Target platforms:
  - Windows
  - Windows Phone
  - Xbox 360
- ❑ Free
- ❑ http://create.msdn.com



```
            if (paddle2.pos.Y <= 10) paddle2.pos.Y = 10;

            if (paddle1.pos.Y + t_paddle1.Height >= Window.ClientBounds.Height - 1
            if (paddle2.pos.Y + t_paddle2.Height >= Window.ClientBounds.Height - 1
        }

        void UpdateBall()
        {
            //update positions
            ball.pos.X += ball.h_speed;
            ball.pos.Y += ball.v_speed;

            //check for boundaries
            //bottom
            if (ball.pos.Y > (Window.ClientBounds.Height - 10 - t_ball.Height))
                ball.v_speed *= -1;

            //top
            if (ball.pos.Y < 10)
                ball.v_speed *= -1;
        }

        void ResetGame()
        {
            paddle1 = new Paddle(10, 200);
            paddle2 = new Paddle(770, 200);
            ball = new Ball(385, 285);
```

# Unity 3D

- ❑ A complete game engine for 3d games that features:
  - ▸ Physics engine
  - ▸ Camera control
  - ▸ Animation system
  - ▸ …
- ❑ Allows development on Windows and Mac both in C# and in Javascript
- ❑ Target platforms:
  - ▸ Windows and Mac
  - ▸ Mobile (Android, iOS)
  - ▸ Web
  - ▸ Consoles
- ❑ Offers free licenses with limitations and paid licenses
- ❑ [http://www.unity3d.com](http://www.unity3d.com)



# Conclusions

# HOT topics / Current Trends

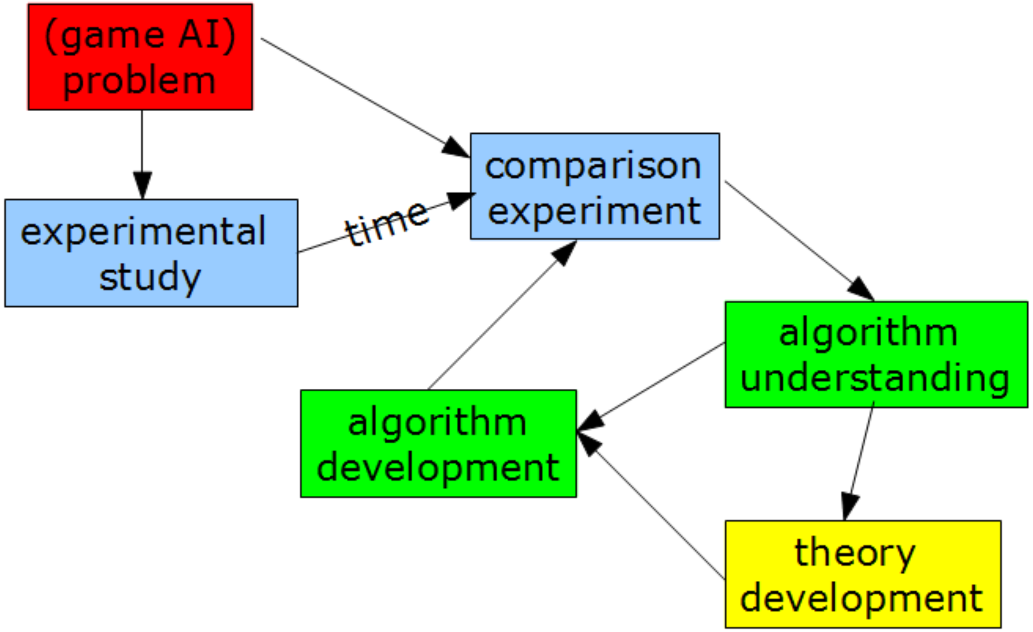
- Believability of NSC and their environment
  - More humanlike behavior
  - Better cooperation of units (team AI)
  - Reactivity to unforeseen events
- Personalization of games
  - Preference modeling (what do they like?)
  - Player type analysis, classification
  - Dynamic adaptation of game mechanisms (e.g. difficulty)
- Procedural Content Generation
  - Offline to support game creation
  - Online to enlarge worlds

# Where are we going?

- ❑ The testbed argument seems to loose importance:
  - test problem collections (benchmarks) and competitions are getting popular in many fields (e.g. BBOB)
  - not really simple to transfer back obtained knowledge (games research partly engineering)
  - the need to defend games research is shrinking
- ❑ The doing things better argument is (still) important:
  - Can involve theory, but usually based on experimentation
  - Question: what does better mean?
  - Measurement sometimes fully automated, sometimes requires user interaction (no fun formula)
  - Required: being open to other methods (to achieve meaningful comparisons)
  - Ideal situation: competition as joint effort experiment (fair)

# To boldly go…

- We may encounter unsolved or even unrealized problems:
  - Interesting features of CI techniques: surprising solutions, highly adaptable to problem, multiple objectives
  - Show that our approach indeed does fulfill some minimal requirements by experiment



# Algorithm development and theory

- ❑ Of course we can improve our methods while applying them
- ❑ But this is usually not unique for games problems
- ❑ Improvement/improved understanding may result in better theory
- ❑ Discrete state games: algorithm engineering cycle applicable
- ❑ More complex games (e.g. RTS): theory connection very difficult
- ❑ Solving games problems is to a large extent engineering
- ❑ *We have to rely on good experimentation in most cases*

# Takehome message

- ❑ CI tools, especially Evolutionary Algorithms are well suited for many applications in Games
- ❑ Multi-objective EA very useful, but rarely used
- ❑ Some very dynamic areas identified: PCG, Personalization
- ❑ Lots of possibilities to enter the arena: competitions, free engines, etc.
- ❑ Game based benchmarks good to motivate people (students) and to showcase your research

- ❑ Newly established web base for CI/AI game research projects and demos: http://www.aigameresearch.org/

Thank you!