UNIVERSITÀ DEGLI STUDI
DI MILANO

Applications of Genetic Algorithms to video games

A.I. for Video Games

Davie Gadia (gadia@di.unimi.it) and Dario Maggiorini (dario@di.unimi.it)

PCG based on evolutionary algorithms

- Goal:
  - to exploit the characteristics of Genetic Algorithms (GAs) to continuously add novel, unpredictable and challenging contents to video games
    - characters (or characteristics of characters)
    - full environments (or specific features)
    - setting up the initial set of features of a level
    - using selection+reproduction+mutation to introduce variety in a ‘’natural’’ way
    - **Quasi-optimal solutions**

Games using GAs: Creatures

- Millennium Interactive (1996)
- Chromosome used to generate minor variations (e.g. skin colour) in the different generations of Norns, while evolution is based on learning and reasoning algorithms

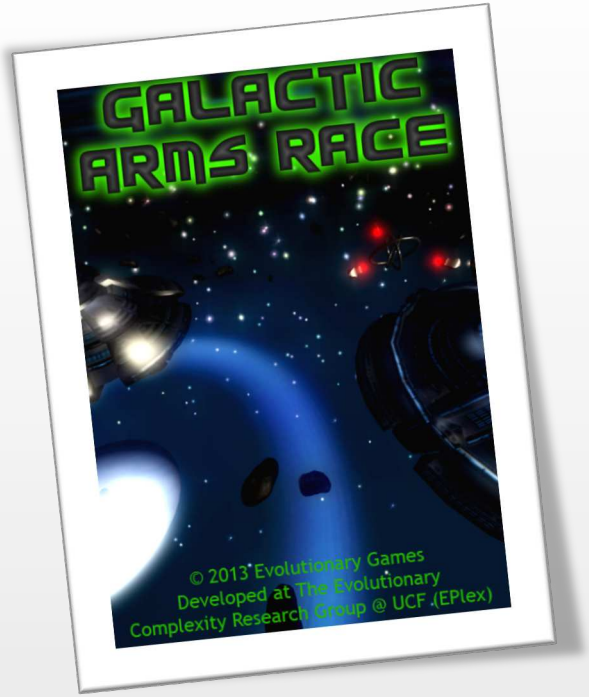
Games using GAs: Spore

- Maxis (2008), designed by Will Wright
- procedural generation on content pre-made by the developers
- the **DNA template of the creature is shared, while the game builds the specific phenotypes**

Games using GAs: GAR

- Evolutionary Games (2010)

- NEAT – **NeuroEvolution of Augmenting Topologies** algorithms to develop weapons accordingly to the player's play style



Games using GAs: From Dust

- Ubisoft Montpellier (2011)
- Uses GAs to create **consequences of «god» intervention** on nature

GAs to add spice to gameplay

GOLEM

GOLEM: Generator Of Life Embedded into MMOs

- MMOs: complex and costly system from both a game design and a technical point of view:
  - They have many open issues, among which gameplay too repetitive
- Goal: incrementing players thrill by supplying plenty of *unpredictable monsters*

Main idea: exploit GAs to make monsters evolve and reproduce in unforeseen species

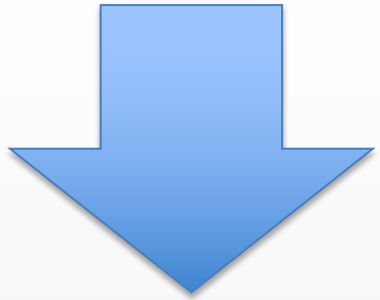
A. Guarneri, D. Maggiorini, L.A. Ripamonti, M. Trubian, “GOLEM: Generator Of Life Embedded into MMOs”.
In Proceedings of the Twelfth European Conference on the Synthesis and Simulation of Living Systems: Advances in Artificial Life, ECAL, pp. 585–592, 2013

GOLEM algorithm

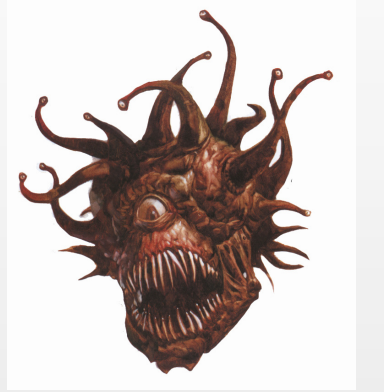
- For each generation (till max.):
  1. **Selection**:
     - Death by old age?
     - Selects candidates and evaluates «challenge ratings» (low/medium/high/invincible)
  2. **Crossover**:
     - Uniform crossover
  3. **Mutation**:
     - Random change of ONE gene (90% probability to happen!!)
  4. **Evaluation**
     - Are whelps suited for their habitat? (death/malus/ etc.)
  5. (**ending**)

GOLEM: Describing monsters

- MMOs set in many different periods and locations:
  - To be «general» enough we have selected *fantasy* setting (the most diffused)



- Genes in the chromosome represent **characteristics** and **skills** derived from the analysis of 150 D&D monsters
  - Immortals and undeads excluded



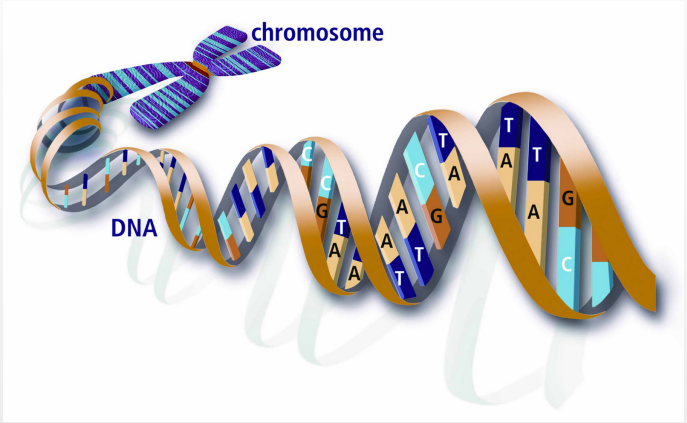
DUNGEONS & DRAGONS

GOLEM: Monsters' chromosome

- Each monster is described by a chromosome:
  - Physical abilities
  - Natural abilities
  - Magic abilities

  37 genes
  - 16 more characteristics mapping skills and other necessary data (e.g., generation number, life, etc.)

- Genes can be clustered in different groups:
  1. must be different between parents (e.g., genre)
  2. necessary to survive (type of breathing, heads, etc.)
  3. optional (resistance to spells, lycanthropy, etc)



GOLEM: Monsters' chromosome: physical, natural, magic

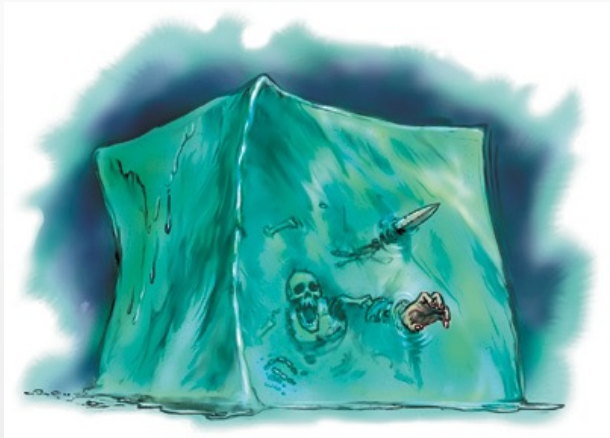
Characteristic	Type	Range
Sex	physical	2
Head number	physical	0-7
Arms number	physical	0-8
Legs number	physical	0-8
Eyes number	physical	0-8
Skin colour	physical	0-8
Eyes colour	physical	8
Tail number	physical	0-8
Wings	physical	y/n
Fins	physical	y/n
Gills	physical	y/n
Scales (fish-like)	physical	y/n
Scales (dragon-like)	physical	y/n
Feather	physical	y/n
Hair/fur	physical	y/n
Size	physical	8
Type (animal, magical creature, etc.)	nat. ab.	8
Movement: swimming	nat. ab.	y/n
Movement: flying	nat. ab.	y/n
Movement: digging	nat. ab.	y/n
Movement: climbing	nat. ab.	y/n

Characteristic	Type	Range
Breathing: air	nat. ab.	y/n
Breathing: water	nat. ab.	y/n
Breathing: fire	nat. ab.	y/n
Breath and type (e.g. fire, ice, etc.)	mag. ab.	6
Natural weapons and type (e.g. claws)	mag. ab.	8
Aura and type (e.g. fire, etc.)	mag. ab.	7
Casts spells	mag. ab.	y/n
Immunity and type (e.g. to poison)	mag. ab.	6
Lycanthropy	mag. ab.	y/n
Shapeshifter	mag. ab.	y/n
Fast healing	mag. ab.	y/n
Regeneration	mag. ab.	y/n
Resistance to spells	mag. ab.	y/n
Resistance to dispel	mag. ab.	y/n
Damages reduction (e.g. thick skin)	mag. ab.	y/n
Poisonous	mag. ab.	y/n

GOLEM: Monsters' chromosome: other characteristics

Max. generations
Current generation
Challenge rating
Life (hits)
Strenght
Dexterity
Constitution
Intelligence
Wisdom
Charisma
Armour class
Speed
Attack
Reflexes
Temper
Will

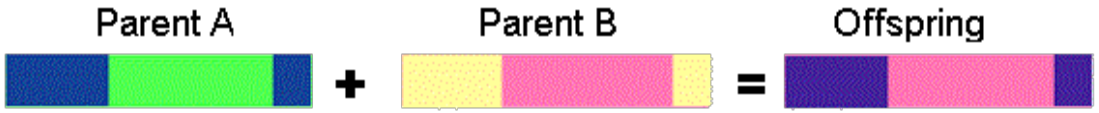
- Each characteristic is described by a number
  - NB: Several D&D characteristics are not significant (hence excluded): e.g. horse-riding



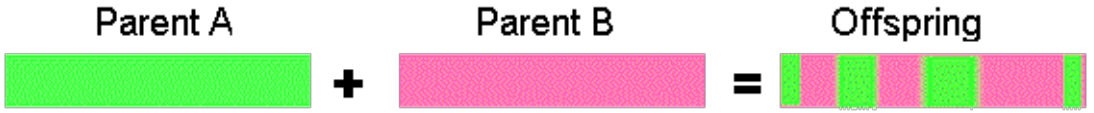
GOLEM: Crossover

- Single point

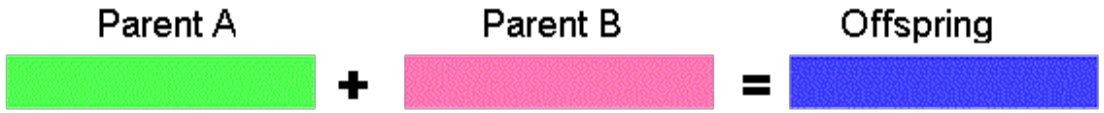
- Two points



- Uniform



- Arithmetic



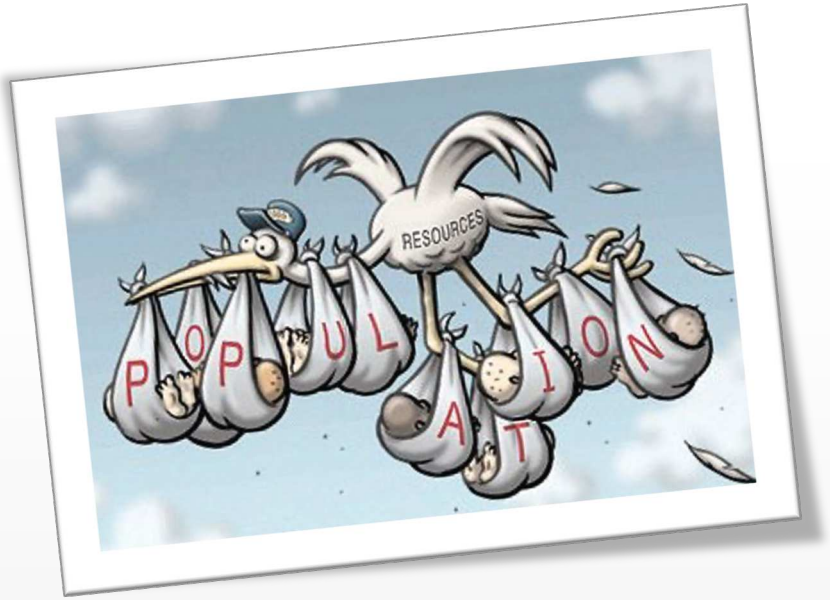
GOLEM: Mutation

- Insertion into the offspring's chromosome of some characteristic not inheritable from parents:
  - can introduce new characteristics or modify/destroy existing ones

- we have considered only the possibility to **create/destroy** characteristics, no (incremental) modify is possible:
  - irrelevant in a MMO
  - too hard to model and animate

GOLEM Life and death: balancing the population

- No ”classic” GA optimal solution search
  - No fitness
  - No stopping condition
  - But need to control population growth



- Possible causes of death:
  1. By unsuitability to the environment
  2. By old age
  3. By chance

- **Death by chance** applies when population numerosity reaches a fixed threshold (70% of max)

GOLEM: crossing a Goblin and a Griffin

Characteristic	Goblin	Griffin	Whelp 1	Whelp 2
Max. generations	*20*	*30*	*30*	*20*
Current generation	*1*	*1*	*1*	*1*
Challenge rating	*1*	*4*	*1*	*1*
Life (hits)	*5*	*59*	*59*	*59*
Strenght	*11*	*18*	*11*	*11*
Dexterity	*13*	*15*	*15*	*15*
Constitution	*12*	*16*	*16*	*16*
Intelligence	*10*	*5*	*5*	*10*
Wisdom	*9*	*13*	*9*	*13*
Charisma	*6*	*8*	*6*	*8*
Armour class	*15*	*17*	*15*	*15*
Speed	*9*	*9*	*9*	*9*
Attack	*1*	*7*	*7*	*1*
Reflexes	*1*	*7*	*7*	*7*
Temper	*3*	*8*	*3*	*3*
Will	*0*	*5*	*5*	*0*
Sex	1	0	1	0
Breathing: air	1	1	1	1
Breathing: water	0	0	0	0
Breathing: fire	0	0	0	0
Head number	0000	0000	0000	0000
Arms number	0001	0000	0001	0000
Legs number	0001	0101	0001	0101
Eyes number	0001	0001	0001	0001
Skin colour	100	000	100	000
Eyes colour	100	101	101	100
Tail	0000	0001	0000	0001
Wings	0	1	1	0
Fins	0	0	0	0
Gills	0	0	0	0
Scales (fish-like)	0	0	0	0
Scales (dragon-like)	0	0	0	0

Characteristic	Goblin	Griffin	Whelp 1	Whelp 2
Feather	0	1	1	0
Hair/fur	0	0	0	0
Spells	0	0	0	0
Size	100	100	100	100
Type	111	011	111	011
Movement: swimming	0	0	0	0
Movement: flying	0	1	0	1
Movement: digging	0	0	0	0
Movement: climbing	1	0	1	0
Breath and type	000	000	000	000
Natural weapons	000	001	000	001
Aura and type	000	000	000	000
Immunity and type	000	000	000	000
Lycanthropy	0	0	0	0
Shapeshifter	0	0	0	0
Fast healing	0	0	0	0
Regeneration	0	0	0	0
Resistance to spells	0	0	0	0
Resistance to dispel	0	0	0	0
Damages reduction	0	0	0	0
Poisonous	0	0	0	0

GOLEM: crossing a Goblin and a Griffin

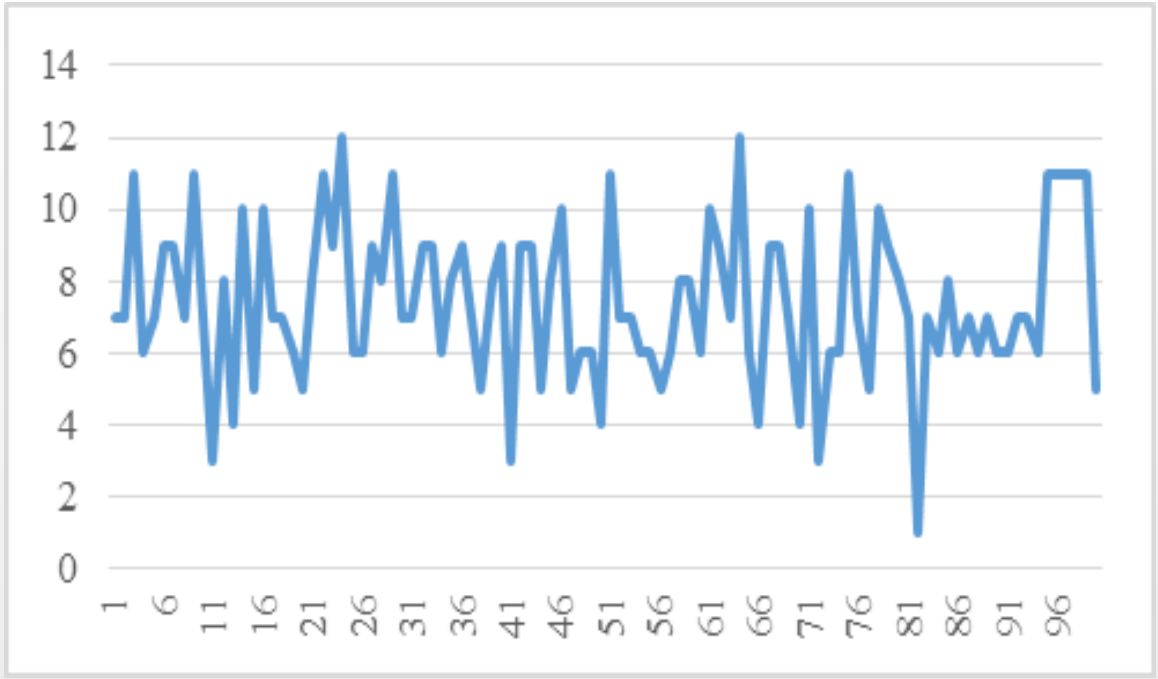
Characteristic	Goblin	Griffin	Whelp 1	Whelp 2
Max. generations	*20*	*30*	*30*	*20*
Current generation	*1*	*1*	*1*	*1*
Challenge rating	*1*	*4*	*1*	*1*
Life (hits)	*5*	*59*	*59*	*59*
Strenght	*11*	*18*	*11*	*11*
Dexterity	*13*	*15*	*15*	*15*
Constitution	*12*	*16*	*16*	*16*
Intelligence	*10*	*5*	*5*	*10*
Wisdom	*9*	*13*	*9*	*13*
Charisma				
Armour class				
Speed				
Attack				
Reflexes				
Temper				
Will				
Sex				
Breathing: air				
Breathing: water				
Breathing: fire				
Head number	0			
Arms number	0			
Legs number	0001	0101	0001	0101
Eyes number	0001	0001	0001	0001
Skin colour	100	000	100	000
Eyes colour	100	101	101	100
Tail	0000	0001	0000	0001
Wings	0	1	1	0
Fins	0	0	0	0
Gills	0	0	0	0
Scales (fish-like)	0	0	0	0
Scales (dragon-like)	0	0	0	0

whelps' genome differs from those of their parents respectively 18.8% and 28.3%

Characteristic	Goblin	Griffin	Whelp 1	Whelp 2
Feather	0	1	1	0
Hair/fur	0	0	0	0
Spells	0	0	0	0
Size	100	100	100	100
Type	111	011	111	011
				0
				1
				0
				0
				000
				001
				000
				000
				0
				0
Fast healing	0	0	0	0
Regeneration	0	0	0	0
Resistance to spells	0	0	0	0
Resistance to dispel	0	0	0	0
Damages reduction	0	0	0	0
Poisonous	0	0	0	0

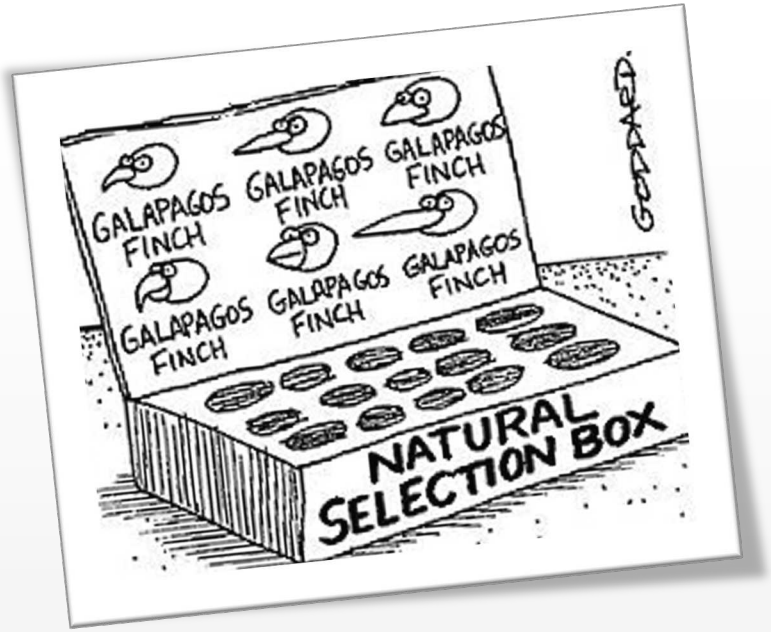
GOLEM Genetic diversity

- Starting population = 18
  - the difference among contiguous generations is of 7.4 genes (13.9%) on average
  - even after 100 generations, the genetic mixing is still substantial



GOLEM Genetic diversity: optimizing performances

- good performance provided by this configuration:
  - population starting numerosity: 18
  - maximum population: 400 individual
  - number of generation: 150
  - number of whelps per generation: 4 or 8



- final numerosity 330, quite good diversity
  - only 10% of the population represented by very similar monsters
  - A further increase in the population numerosity causes the appearance of too many very similar chromosomes

GOLEM-based/inspired projects

- GOLEM GA can be applied and extended to different projects and games
  - different graphical representations
  - different game genres
  - Integrated with AI and/or specific game mechanics
  - …..

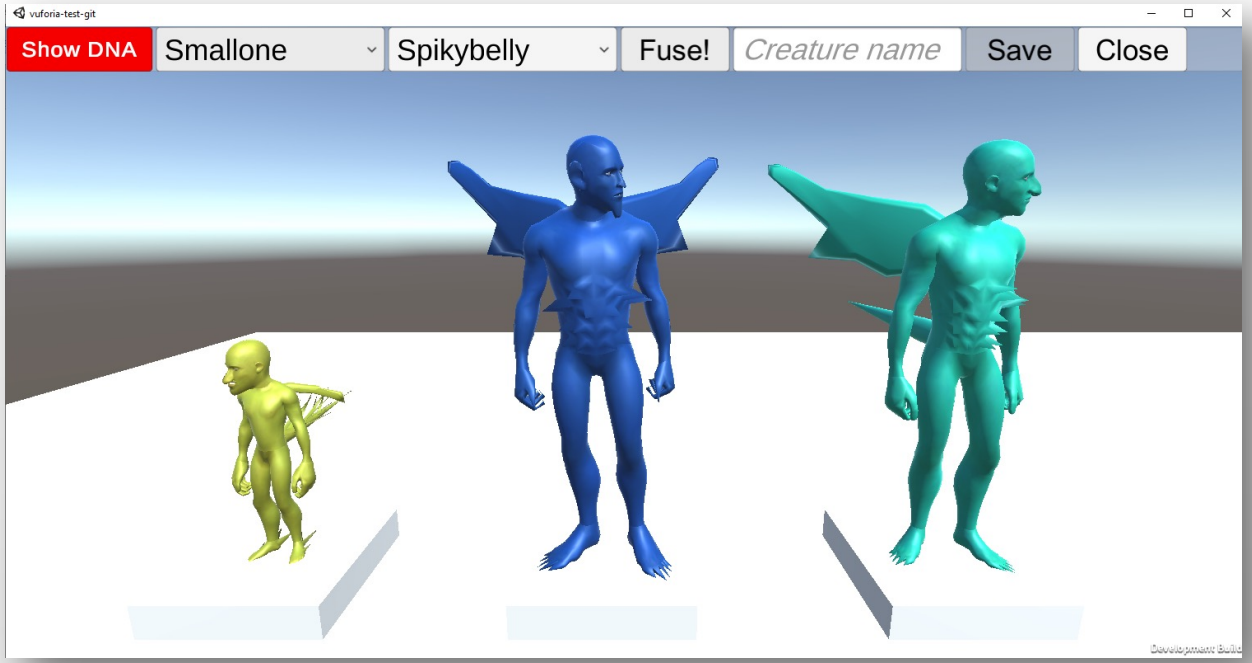
GOLEM-based projects: Find, Fuse, Fight

- AR game inspired by Pokemon GO
- Goal of the game: breed new monsters

- **Find** eggs which contain (procedurally generated) Creatures
- **Fuse** pairs of Creatures to generate a new one that inherits characteristics from both
- **Fight** against another player in an online multiplayer, turn-based battles

GOLEM-based projects: Find, Fuse, Fight

- **Fusion** GA extends the one presented in GOLEM
  - More genes for body parts
  - Genes for battle-related statistics
  - Genes for different attack types
  - Different Crossovers and Mutation for the different kind of genes



GOLEM-based projects: Monster of Darwin (MoD)

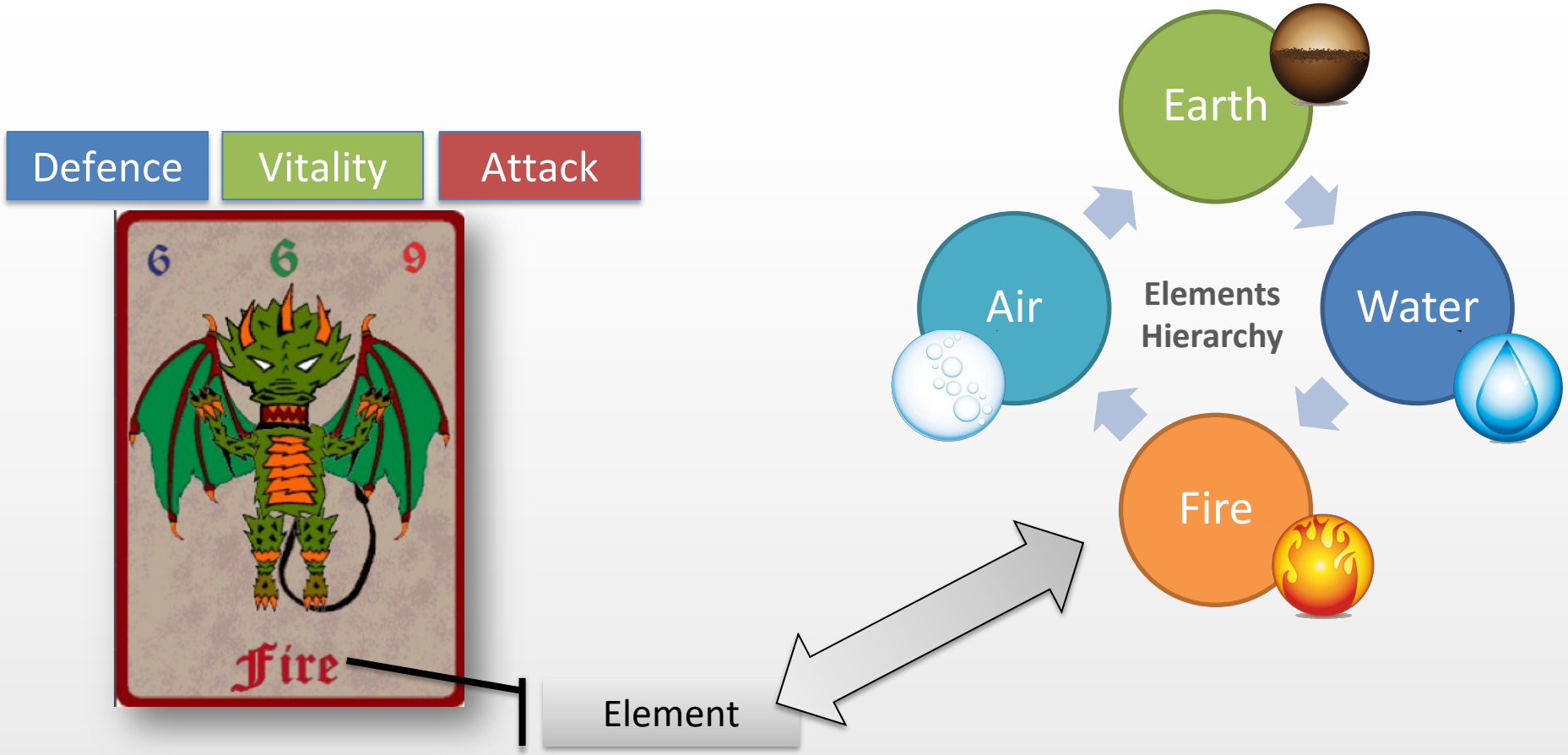
- Turn-based single/multiplayer card-based strategic battle video game for mobile

- Goal: design and prototype a dynamically changing Trading Card Game supported by AI
  - **Minimax** algorithm to implement the strategic component
  - **GAs** to create diversity and variation among the game objects

D. Norton, L.A. Ripamonti, M. Ornaghi, D. Gadia, D. Maggiorini, “Monsters of Darwin: a strategic game based on Artificial Intelligence and Genetic Algorithms”. GHITALY17: 1st workshop on Games-Human Interaction, Sept. 18th 2017, Cagliari Italy

GOLEM-based projects: Monster of Darwin (MoD)

- Cards inspired by grimoires and medieval manuscripts



GOLEM-based projects: Monster of Darwin (MoD)

- Goal: collecting the largest quantity of different *Monster Cards*

- For each turn:
  - Two cards are played (one by the player on turn, one by the AI)
  - The player on turn decides whether monsters should:
    - *Combat*
    - *Mate*

MoD strategic component

PLAY A CARD

Two cases:

1. **AI goes first**:
   card to play randomly extracted

2. **Player goes first**:
   AI examines player's card and plays a stronger card
   - Natural element which rules
   - Stronger attack

ACTIONS

AI selects mating/combat basing on:

- **Remaining** cards (in deck)
- **Strenght** and **compatibility** between cards in play
- **Vitality gap** between cards in play

Both based on MiniMax

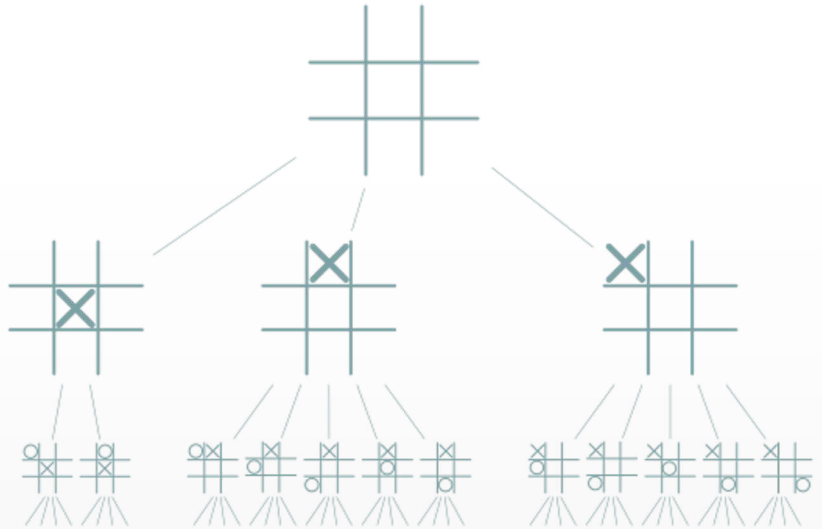
MoD strategic component: Minimaxing

- AI role (in a generic TBS game):
  - Find possible next actions, evaluate them, select next action
    - Pick the best choice
  - Static evaluation function
    - **Minimize opponent score, maximize AI score**

- Recursive algorithm:
  - For each iteration, calculate the current value of game state:
    1. find every possible next move
    2. for each possible next move calculate resulting state value
    3. iterates

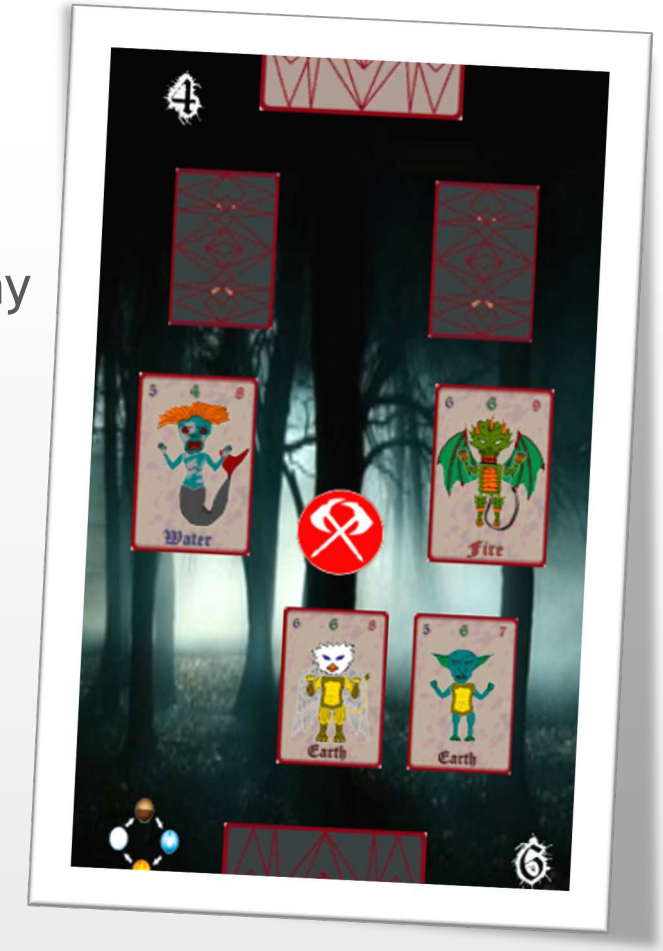
Minimax algorithm

- Recursive algorithm on a tree data structure
  - game tree
    - each node = a board position
    - each branch = one possible move
  - for each iteration
    - the algorithm evaluates the current state
    - considers each possible move and corresponding new board positions
    - recursively calculates the score for each new possible situation, in order to choose the most appropriate one
      - i.e., the one which minimizes the opponent's score
      - static evaluation function



MoD Actions: Combat

- Damage inflicted to Vitality of opponent's card
  - Damage = attack - defence
  - damage increased/decreased (+1, -1) according to Elements hierarchy
  - if Vitality = 0 → card removed from play



MoD mating

- GA derived by GOLEM's GA
  - generates a new monster starting from the pair of cards in play
  - initial set of 8 cards

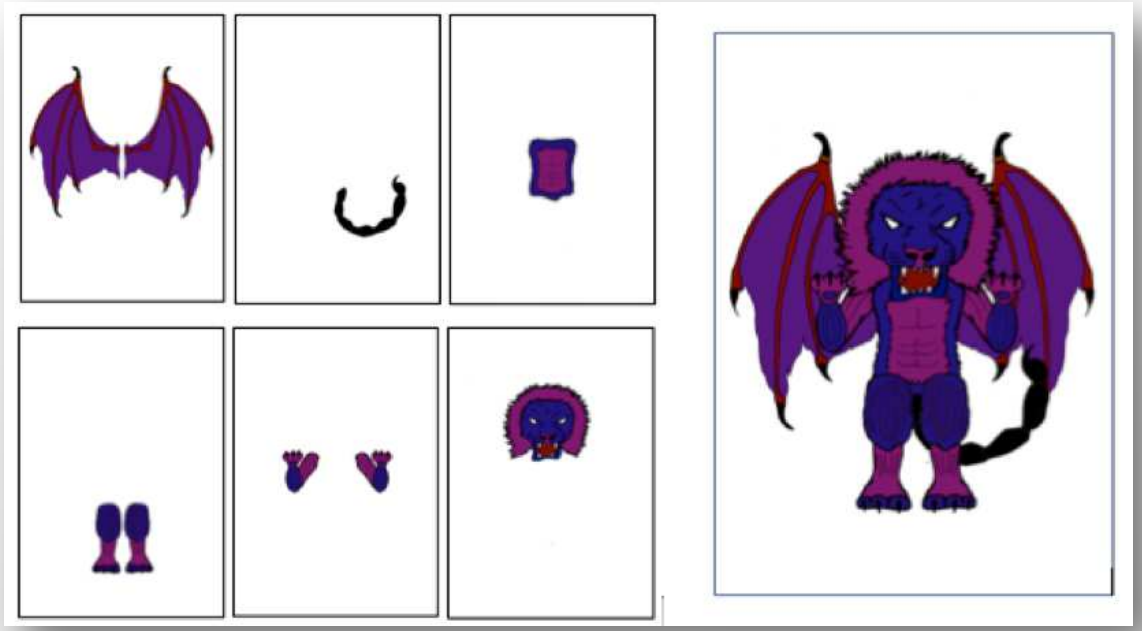
MoD mating

- Monsters represented by chromosomes:
  - Physical traits
  - Strenght
  - Natural element

Characteristics	Type	Range
head	physical	8
eyes	physical	8
arms	physical	8
body	physical	8
legs	physical	8
tail	physical	8
wings	physical	8
attack	strenght	3
defense	strenght	3
vitality	strenght	3
element	natural element	4

MoD physical traits

- Each physical trait refers to its correspective in one of the 8 original Monster Cards
- Each Monster picture is composed by textured 2D patches:
  - New monsters are composed by mixed patches from parents' cards



MoD physical traits

Compatibility: evaluated on the couple of Monsters:

- the element of the card belonging to player on turn MUST **rule over the other**
- difference in Vitality MUST be **>= 2**
- if not compatible:
  - the card of the player on turn receives a **damage**
- if compatible:
  - **Uniform crossover** is applied to provide the highest variability

GAs to add variety in NPCs behaviour

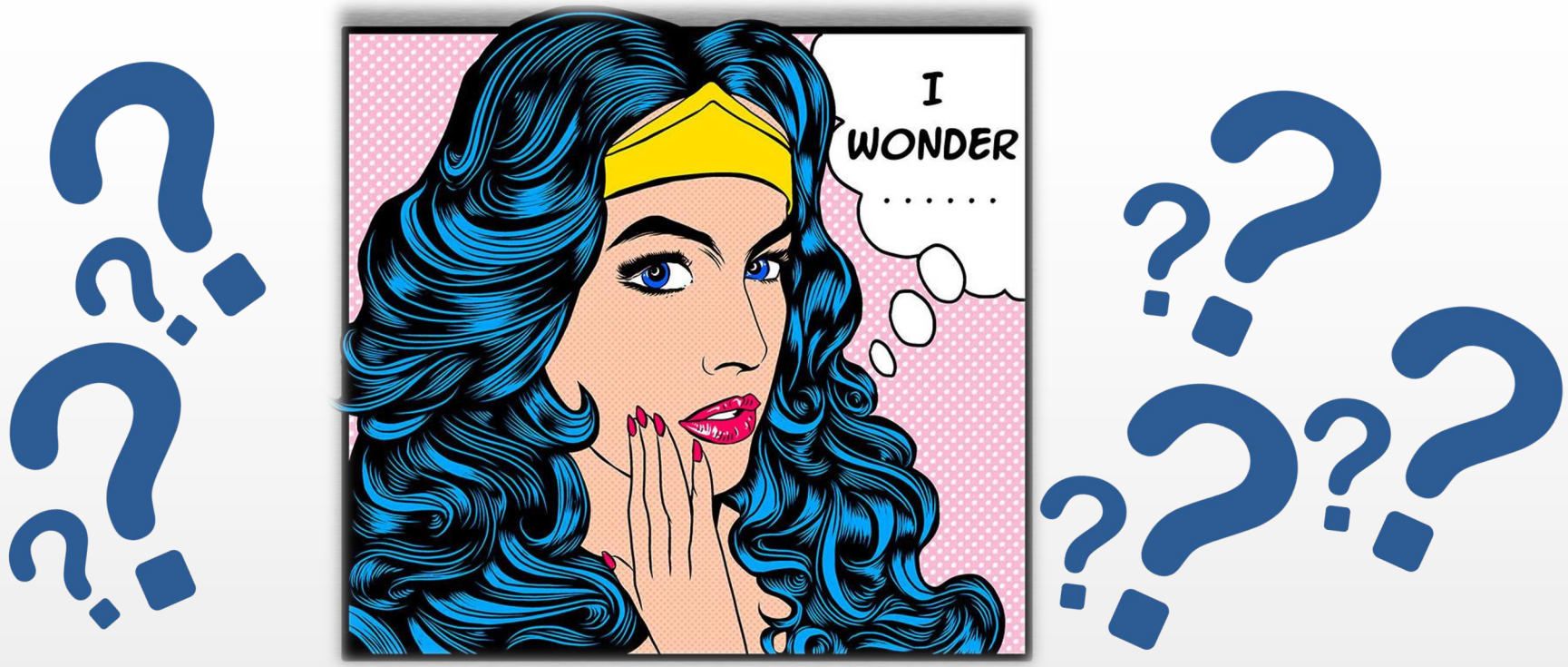
GENIE

Our research question

Is it possible to find an easy, quick and cheap way to generate **not trivial and not predictable** NPCs (Non-Player Character)?

Wait a minute …

- … is unpredictability really useful in games ???



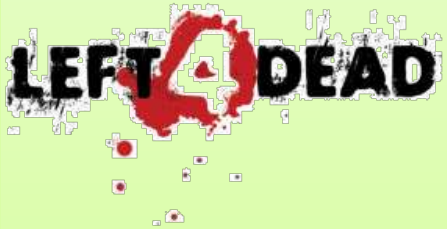
… well, it depends …

Competitive games

- Players MUST face the same situation

- Unpredictability negatively affects:
  - Simmetry
  - Fun
  - Skill growth

Single player or cooperative



Approach rooted into «emotion» simulation

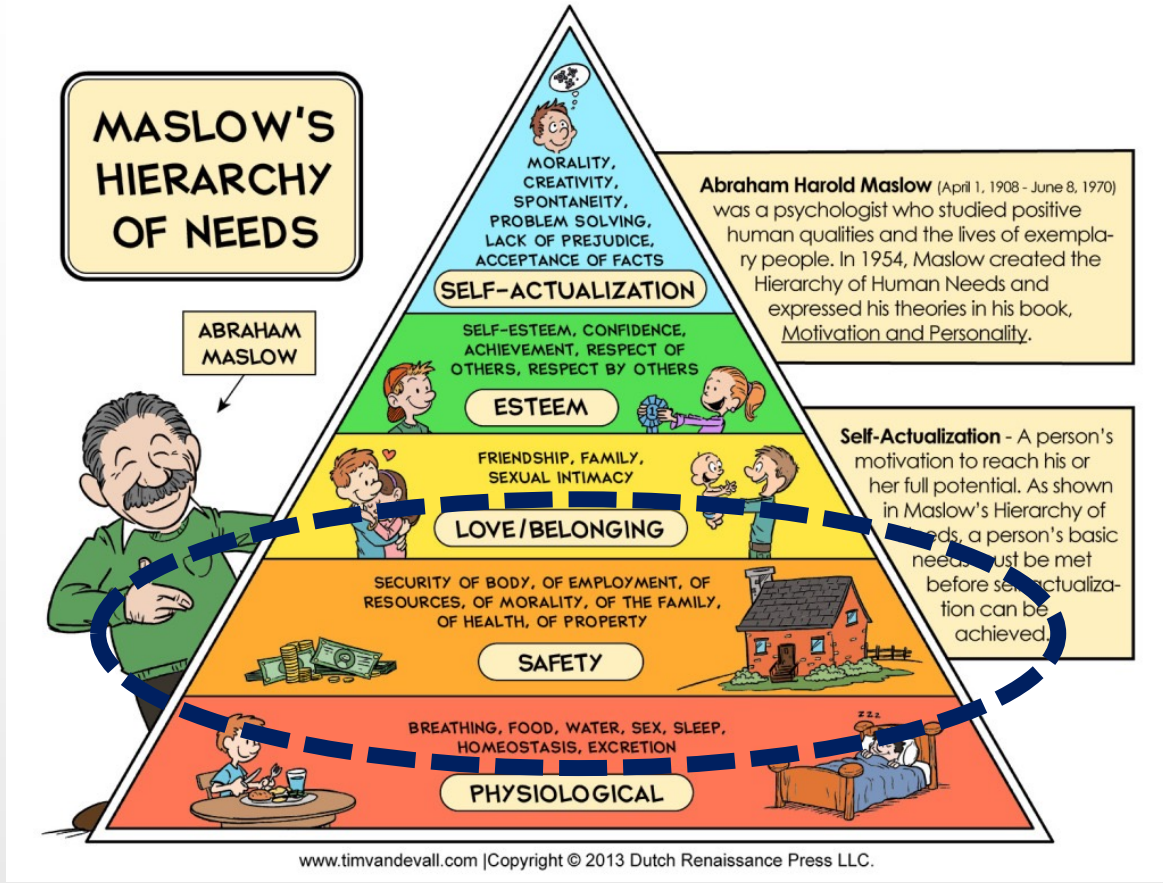
Adding «emotions»

- Illusion of life
- Increased credibility
- Increased unpredictability

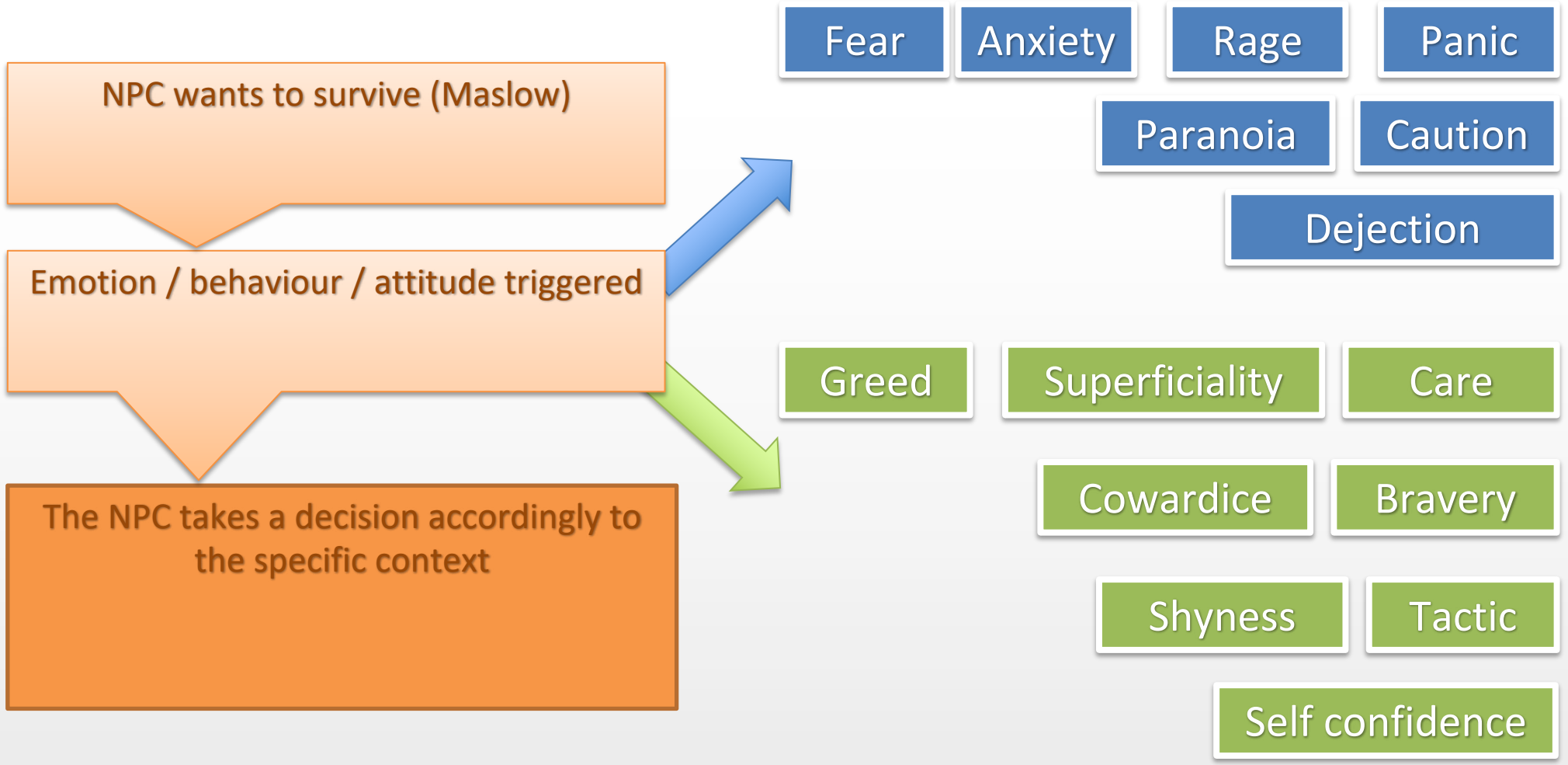
Which emotions?

- We need to analyse the NPC behaviour in its context (game)

Typical goal:
survive and **kill the player**



Which «emotions»?



GENIE – Genes Driven Decision Tree

plug-in for Unity3D that forces some ***emotionality*** *on NPCs to add some «spice» to their behaviours*

Creation of a *Decision Tree*

Decisions weighted by «emotional» parameters

Parameters generated by a GA and evaluated

F. Agliata, M. Bertoli, L. A. Ripamonti, D. Maggiorini, D. Gadia, "Adding Variety in NPCS Behaviour using Emotional States and Genetic Algorithms: The Genie Project", Proceedings of GAME-ON 2019 conference, pp. 45-50, 2019

Difference with other approaches in literature

Application of GA or generative approaches to NPC is not a complete novelty, BUT:

Generative behaviour might be too different from previous one

Loss of continuity

Evolution approaches often aimed at outperforming the human player

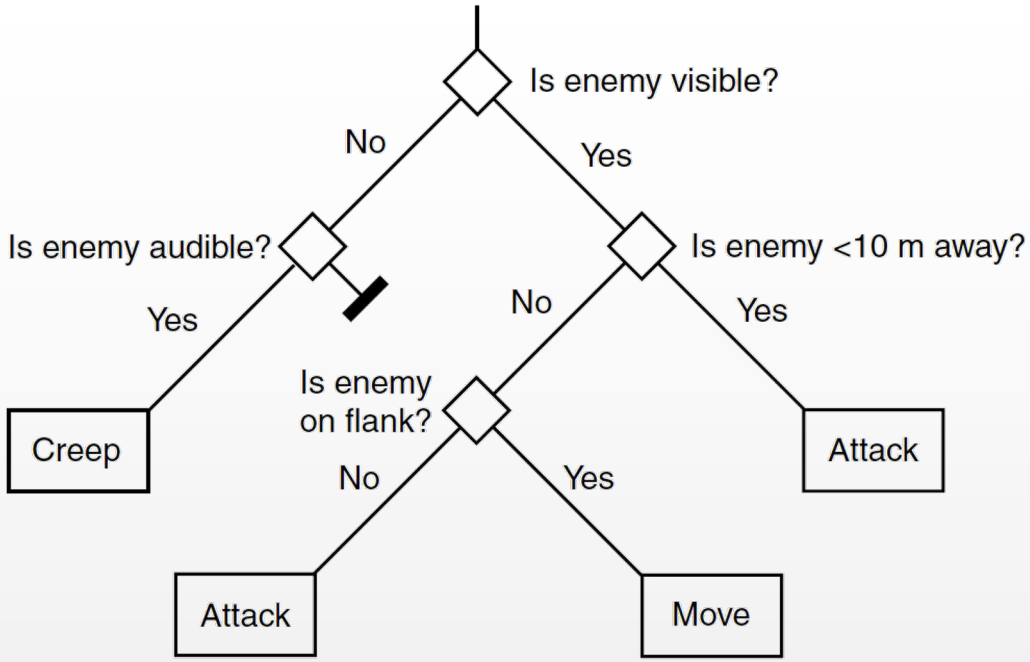
Reasonable for racing or chess, poor choice for plot-driven games

In our vision: evolution should provide continuous and smooth changes to follow the player skill and keep her in the game flow

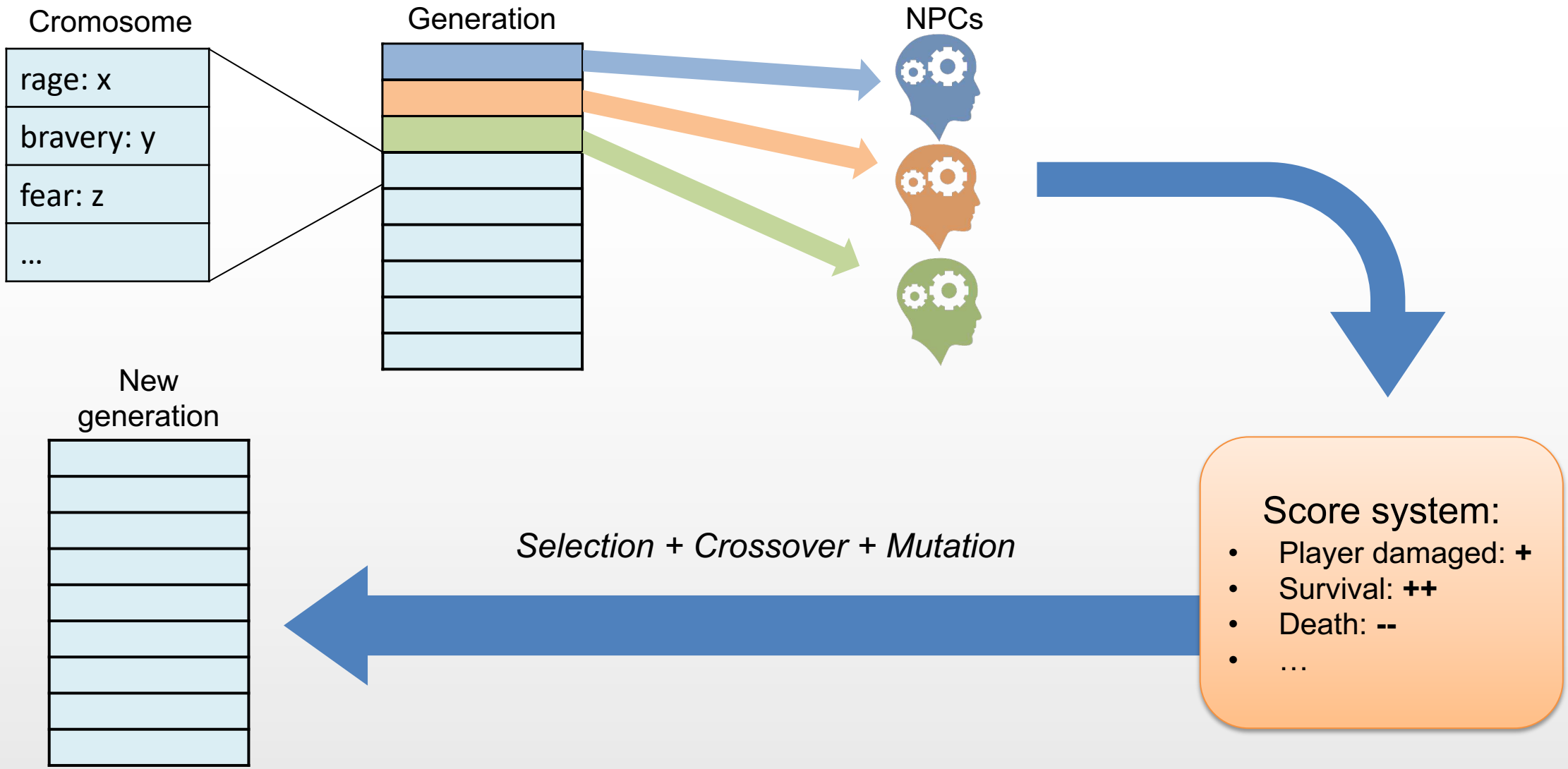
GENIE core idea

Decision trees (binary)

- Easy to design
- Good performance
- Easy to implement



GENIE core idea: «emotion (generation)»



«Emotions» representation

Real [0;1]

Rage = 1.0

Greedy = 1.0

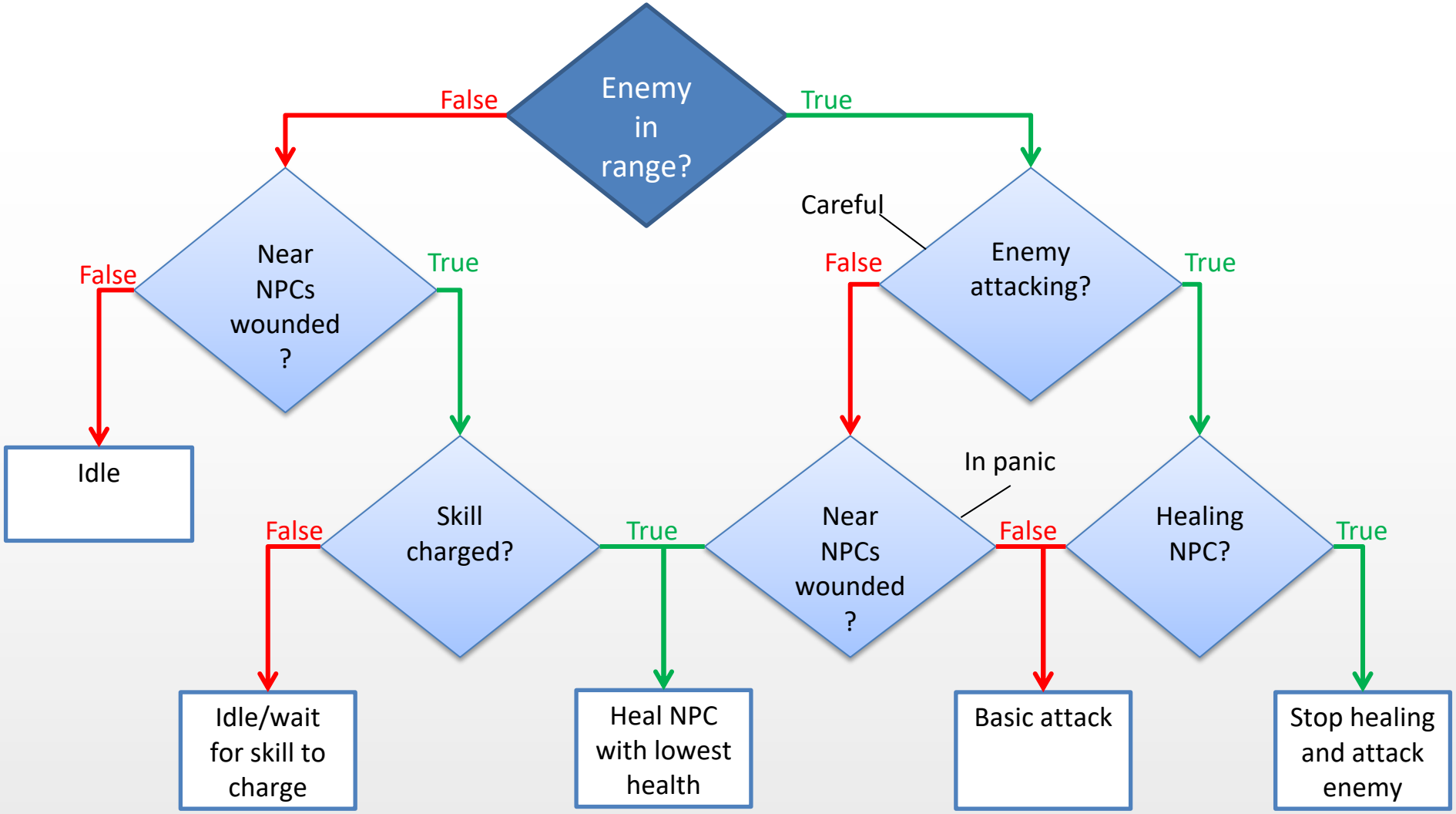
Rage = 1.0
Greedy = 1.0

Example: NPCs enraged or scared

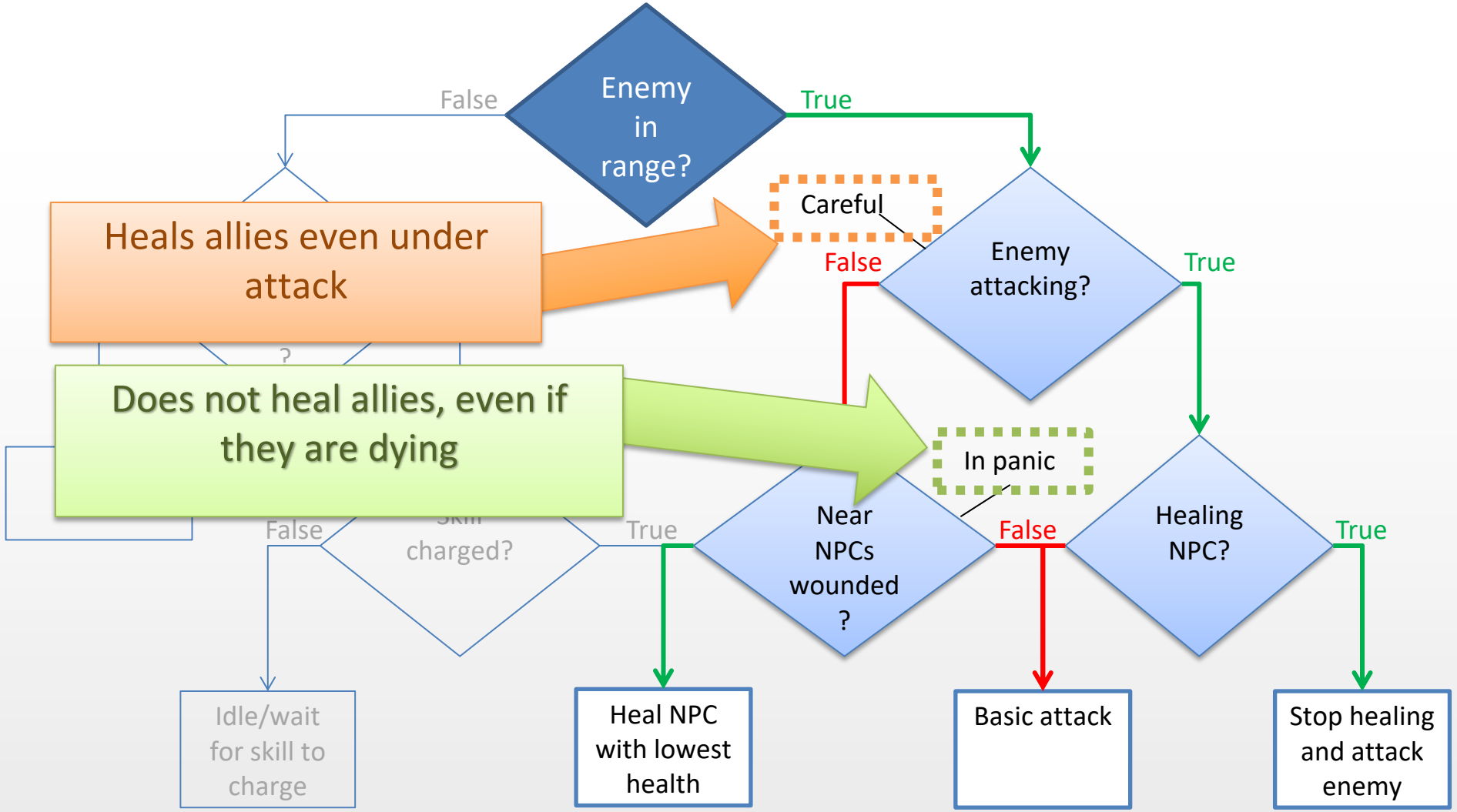
Rage (action FPS)		
NPCs attack player even without weapons	NPCs do not flee, even if heavily wounded	NPCs do not hide

Anxiety/fear (RPG)	
Melee	Healer
NPCs always flee from player	NPCs do not heal allies, even when not under attack
NPCs always try to find shelter near allies	

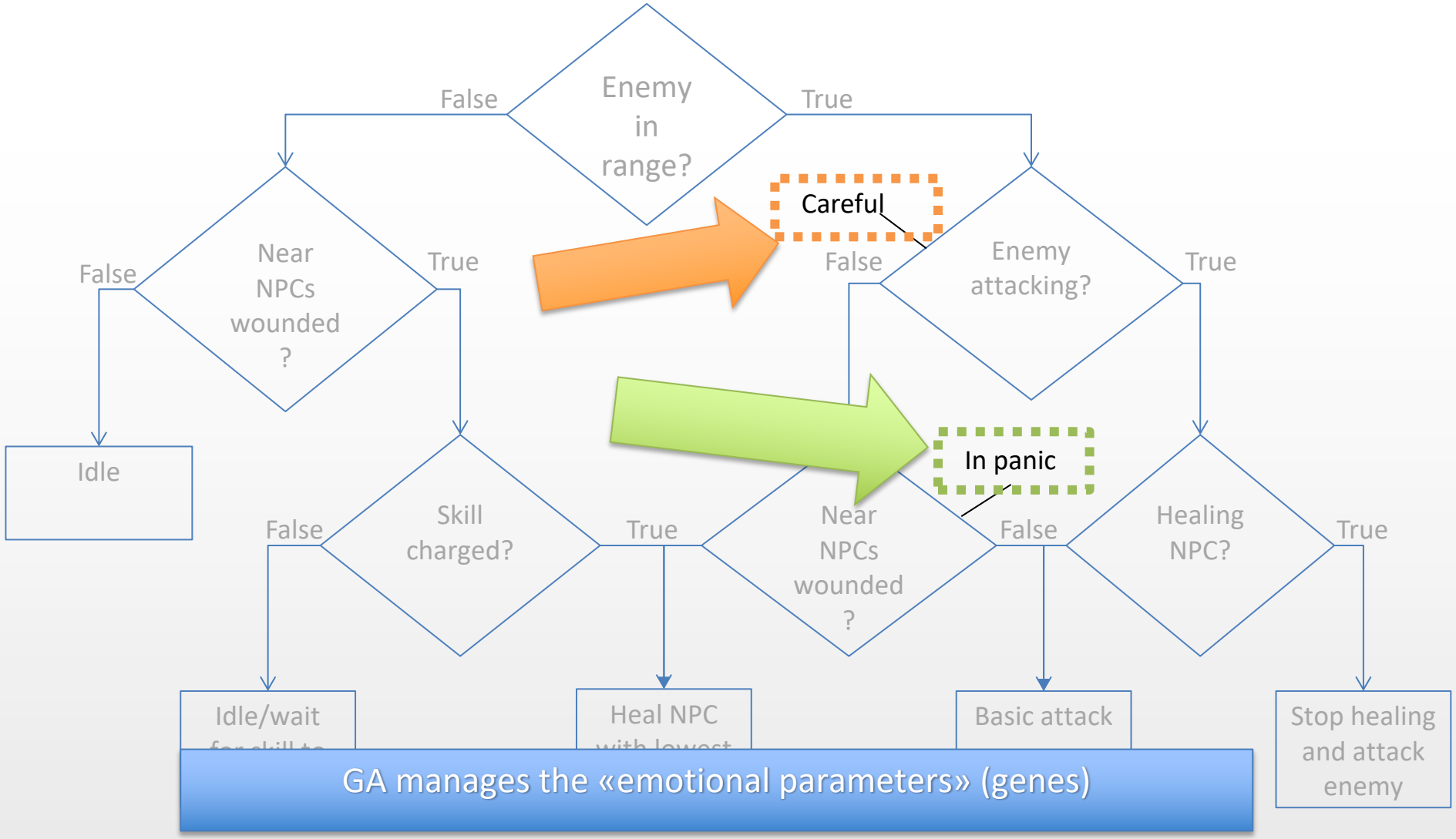
«Emotional decisions» - NPC Healer (RPG)



«Emotional decisions» - NPC Healer (RPG)



«Emotional decisions» - NPC Healer (RPG)

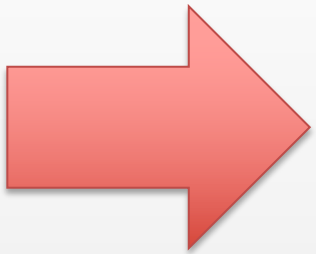


GA: scoring System & fitness function

single-point crossover

Random mutation of a randomly chosen chromosome

Fitness function



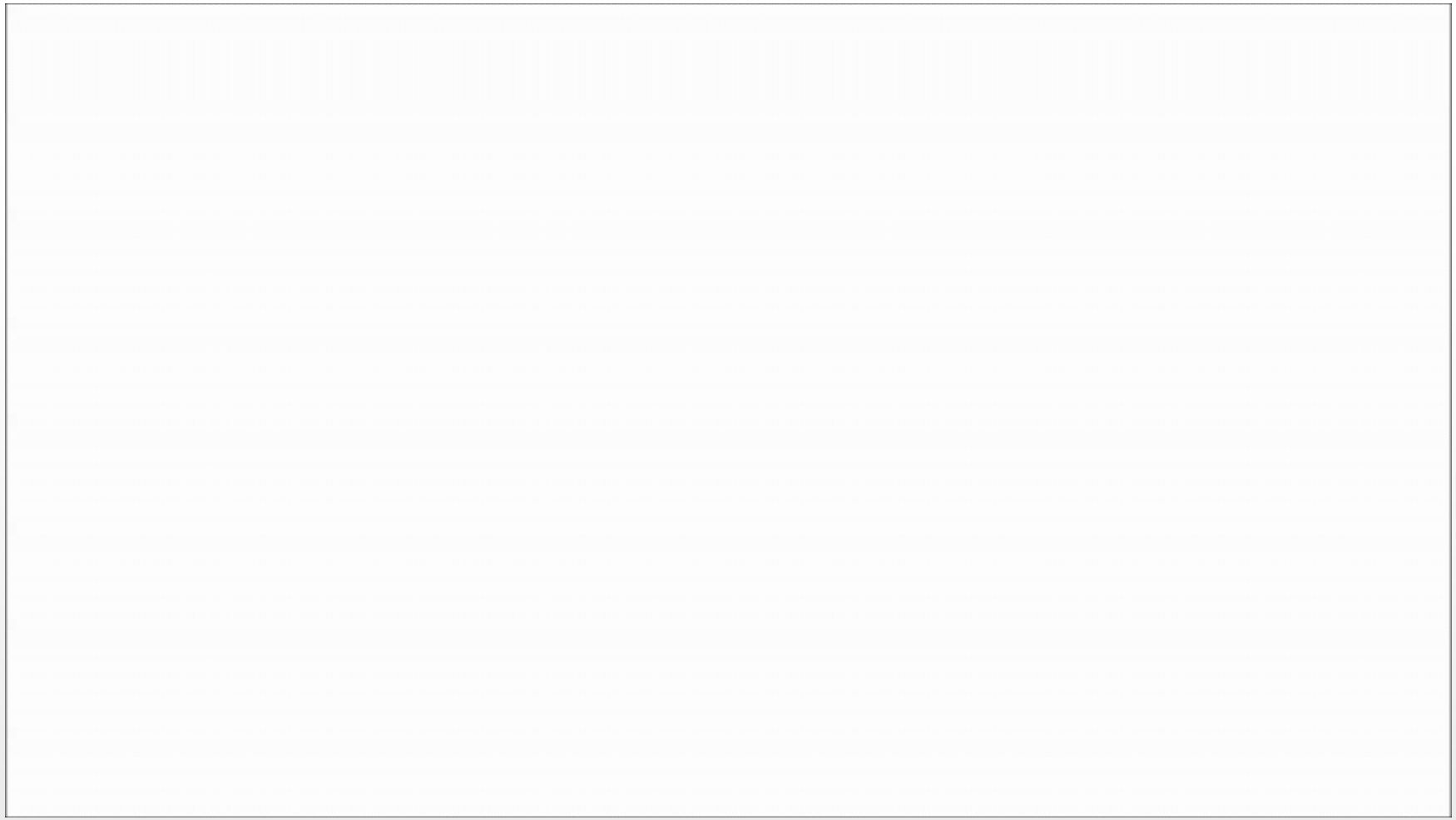
Based on a scoring system:

- evaluates each NPC effectiveness
- Parameters must be customized for each game

Scoring example: Action FPS

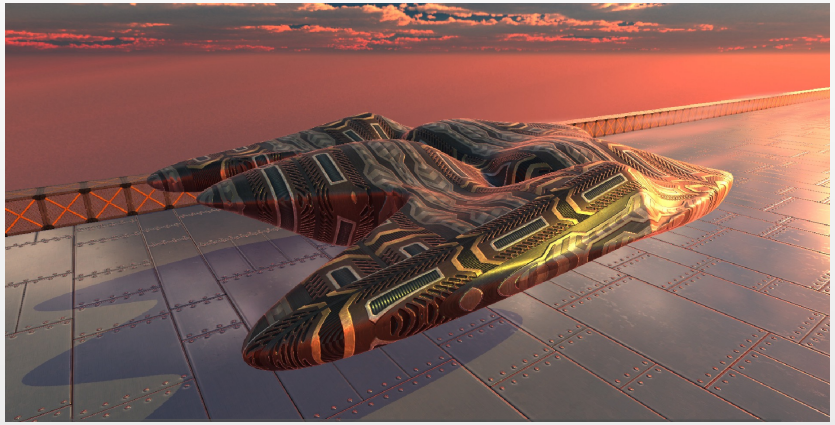
Action FPS scoring		
Player killed	+2	
Ranged damage dealth	+2	
Melee damage dealth	+1	
Damage dealth (throwing weapon)	+1	
Healing (self)	+1	
Weapon collected	+1	
Time interval (t) before death	t < 5sec.	-1
	5 <= t < 7	0
	7 <= t < 9	+1
	t >= 9	+2

Examples: Action (Rage) / RPG (Anxiety)



GAs to introduce/control variety
in game assets appereance

Evolving materials and spaceships

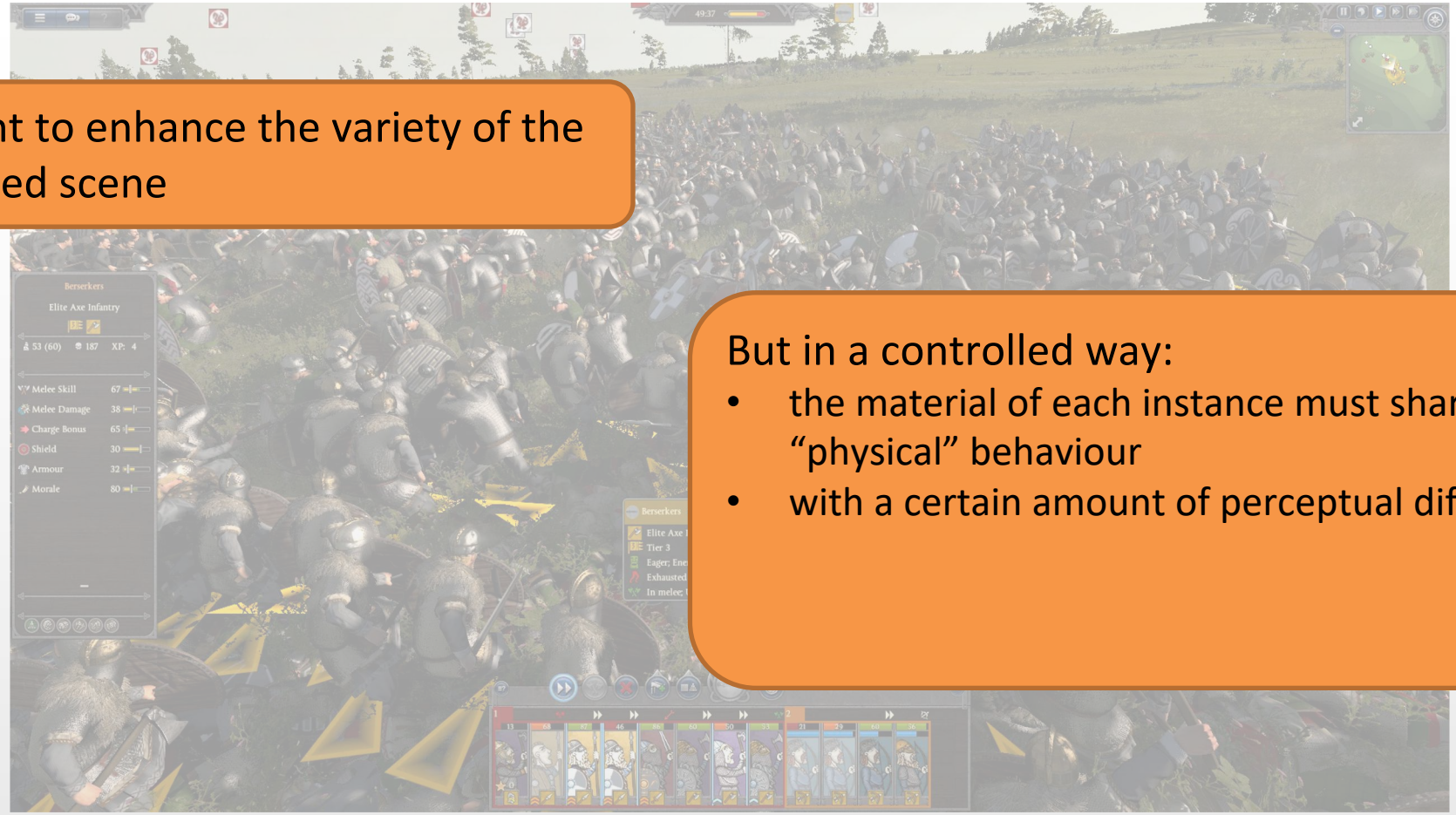


PCG of realistic materials for Real-Time Rendering

- Often, several instances of a model are generated in a large virtual world

PCG of realistic materials for Real-Time Rendering

- Often, several instances of a model are generated in a large virtual world



We want to enhance the variety of the generated scene

But in a controlled way:

- the material of each instance must share a common "physical" behaviour
- with a certain amount of perceptual differences

PCG of realistic materials for Real-Time Rendering

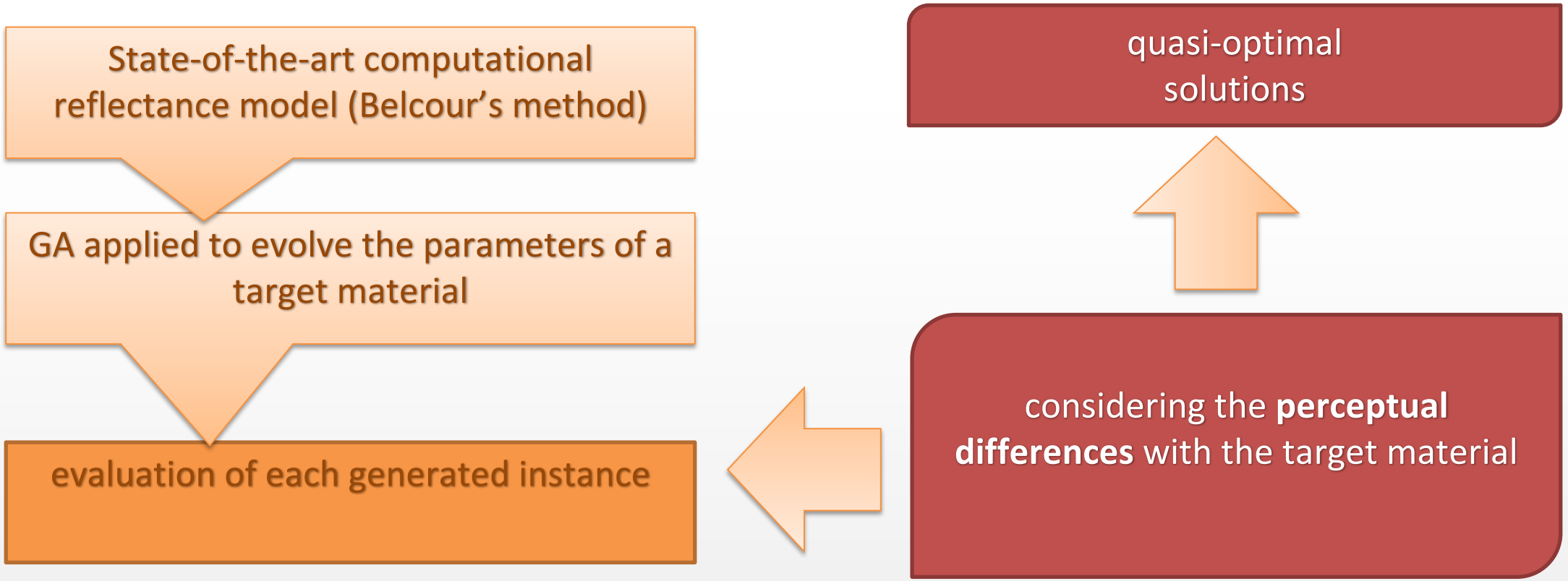
State-of-the-art computational reflectance model (Belcour's method)

GA applied to evolve the parameters of a target material

evaluation of each generated instance

A. Bernardi, D. Gadia, D. Maggiorini, C. E. Palazzi, L. A. Ripamonti, "Procedural Generation of Materials for Real-Time Rendering", Springer Multimedia Tools and Applications, 2020

PCG of realistic materials for Real-Time Rendering

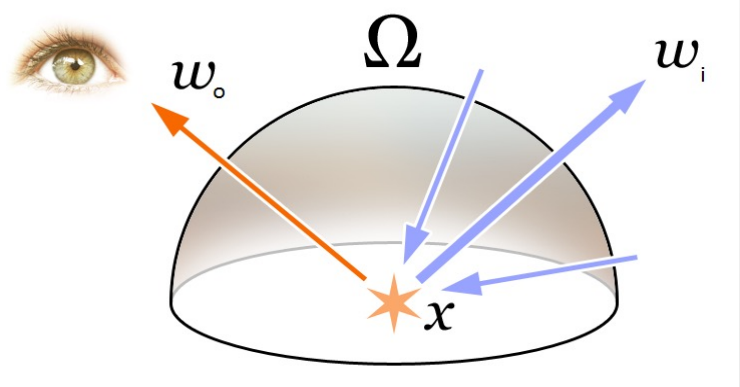


A. Bernardi, D. Gadia, D. Maggiorini, C. E. Palazzi, L. A. Ripamonti, "Procedural Generation of Materials for Real-Time Rendering", Springer Multimedia Tools and Applications, 2020

But first of all, some background on CG rendering

- Simulation of physical interaction of light and material
- CG rendering equation :

$$L_o(\vec{\omega_o}) = \int_{\vec{\omega_i} \in \Omega} f_r(\vec{\omega_i}, \vec{\omega_o}) L_i(\vec{\omega_i})(\vec{\omega_i} \cdot \vec{n}) \partial \vec{\omega_i}$$



But first of all, some background on CG rendering

- Simulation of physical interaction of light and material
- CG rendering equation :

direction of incoming light

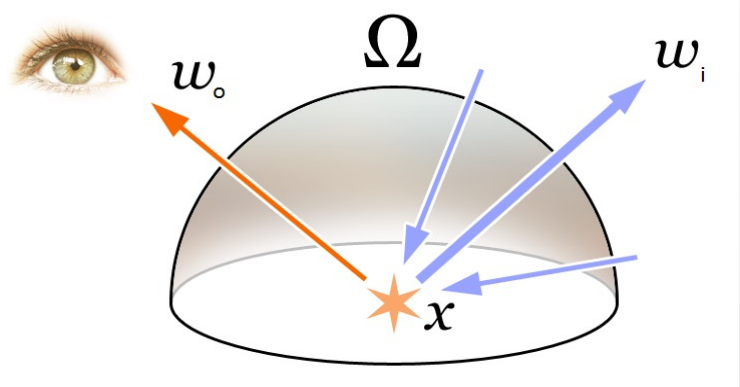
direction of reflected light

surface normal

reflected radiance

$$L_o(\vec{\omega_o}) = \int_{\vec{\omega_i} \in \Omega} f_r(\vec{\omega_i}, \vec{\omega_o}) L_i(\vec{\omega_i}) (\vec{\omega_i} \cdot \vec{n}) \partial \vec{\omega_i}$$

incoming radiance



But first of all, some background on CG rendering

- Simulation of physical interaction of light and material
- CG rendering equation :

direction of incoming light

direction of reflected light

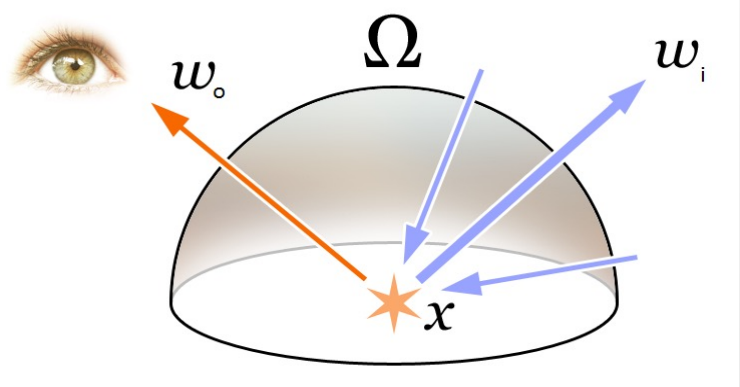
surface normal

reflected radiance

$$L_o(\vec{\omega_o}) = \int_{\vec{\omega_i} \in \Omega} f_r(\vec{\omega_i}, \vec{\omega_o}) L_i(\vec{\omega_i}) (\vec{\omega_i} \cdot \vec{n}) \partial \vec{\omega_i}$$

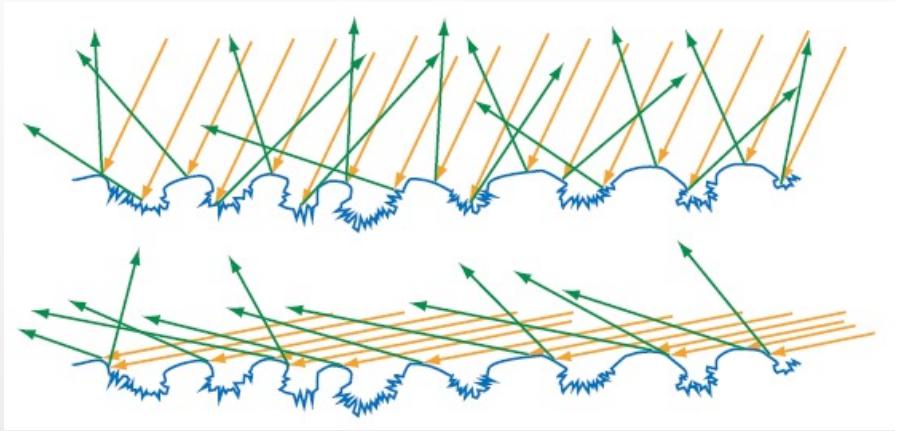
incoming radiance

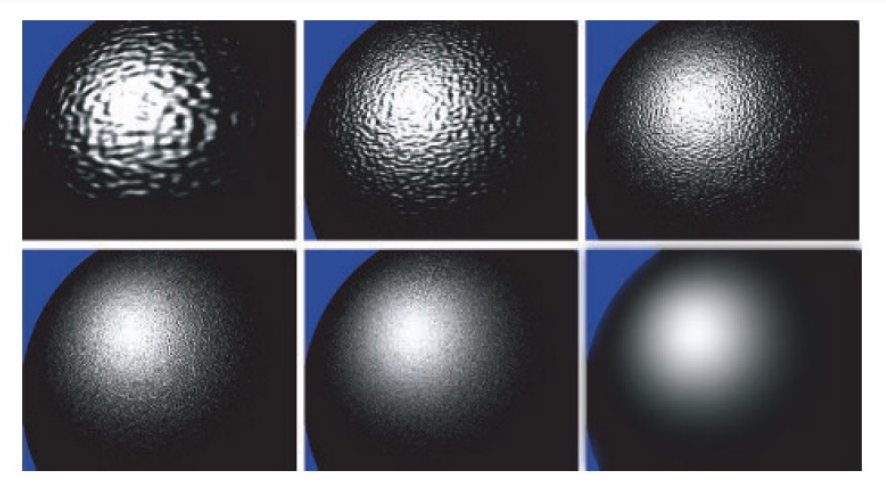
Bidirectional Reflectance Distribution Function (BRDF)



But first of all, some background on CG rendering

- In real-time CG
  - coarse approximation of rendering equation
    - BRDF approximated using *microfacets approach*
    - scattering properties of a material described by the *statistical distribution of the microfacets orientations*





And what about GAs used to generate BRDFs?

A topic not extensively investigated in literature

And what about GAs used to generate BRDFs?

Brady et al. [2014]

- Genetic programming
- framework to create new analytic BRDFs
- symbolic transformations to two parent BRDFs
- to generate more complex models

Masia et al. [2014]

- Genetic algorithm
- determination of reflectance characteristics of an object in an image
- parameters of two well-known BRDFs are used as chromosomes

Sitthi-Amon et al. [2011]

- Genetic programming
- Simplification of a shader source code

Our approach

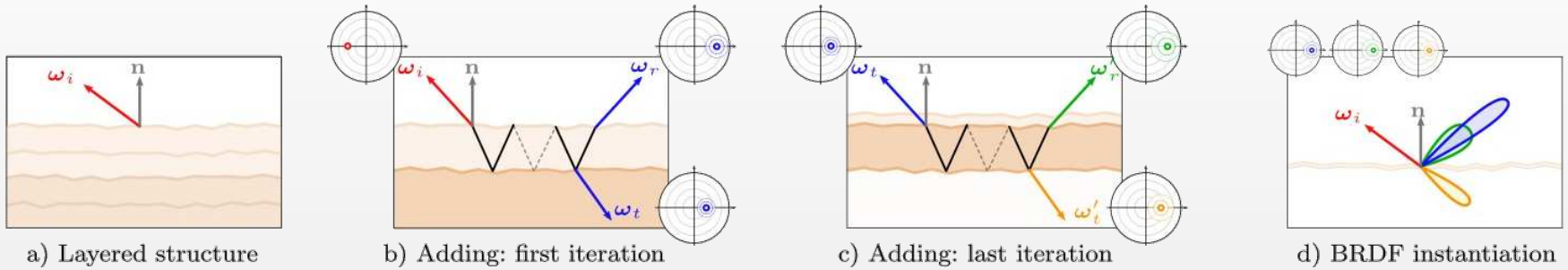
Belcour's method for rendering of layered materials

GA applied to the parameters of Belcour's method to generate a desired amount of moderately different materials

Experimental evaluation

Belcour's method

- Layered materials
  - Each layer has its BRDF
    - usually illumination is computed at each layer
    - blending to simulate internal scattering
  - Belcour proposes an innovative approach
    - atomic statistical operators (from mean, energy, variance at each layer)
    - operators are combined to obtain a single BRDF approximation



a) Layered structure b) Adding: first iteration c) Adding: last iteration d) BRDF instantiation

Laurent Belcour. "Efficient Rendering of Layered Materials using an Atomic Decomposition with Statistical Operators." ACM Transactions on Graphics, 37(4):1, 2018.

Belcour's method

	Rough Reflection	Rough Refraction	Absorption	Forward Scattering
energy	$e^R = e_i \times \mathrm{FGD}^{\infty}$	$e^T = e_i \times [1 - \mathrm{FGD}^{\infty}]$	$e^T = e_i \exp\left[-\frac{\sigma_t h}{\sqrt{1-\lvert\mu_i\rvert^2}}\right]$	$e^T = e_i \left[\frac{\sigma_s h}{\sqrt{1-\lvert\mu_i\rvert^2}}\right] \exp\left[-\frac{\sigma_t h}{\sqrt{1-\lvert\mu_i\rvert^2}}\right]$
mean	$\mu^R = -\mu_i$	$\mu^T = -\eta_{12}\,\mu_i$	$\mu^T = -\mu_i$	$\mu^T = -\mu_i$
variance	$\sigma^R = \sigma_i + \sigma_{12}^R$	$\sigma^T = \frac{\sigma_i}{\eta_{12}} + f(s \times \alpha_{12})$	$\sigma^T = \sigma_i$	$\sigma^T = \sigma_i + \sigma_g$

α is the layer roughness
s is a roughness scaling factor for the trasmission
η_{12} is the ratio of the refractive indices
h is the depth of the layer
σ_t is the transmittance cross-section
σ_s is the scattering cross-section
σ_g accounts for the increase in variance due to the width of the phase function

Proposed GA

- Belcour's method very well suited for GA
  - compact description
  - limited and intuitive parameters
- For Real-Time rendering:
  - 2 layers + possible participating media inbetween

η_1	refractive index of the top layer
η_2	refractive index of the bottom layer
α_1	roughness of the top layer
α_2	roughness of the bottom layer
h	depth of the partecipating media
σ_s^R, σ_s^G , σ_s^B	scattering cross section values
σ_a^R, σ_a^G , σ_a^B	absorption values
g	anisotropic factor

Proposed GA

- Belcour's method very well suited for GA
  - compact description
  - limited and intuitive parameters
- For Real-Time rendering:
  - 2 layers + possible participating media inbetween

η_1	refractive index of the top layer
η_2	refractive index of the bottom layer
α_1	roughness of the top layer
α_2	roughness of the bottom layer
h	depth of the partecipating media
σ_s^R, σ_s^G , σ_s^B	scattering cross section values
σ_a^R, σ_a^G , σ_a^B	absorption values
g	anisotropic factor

If no participating media, only these 4 control final appearance

Proposed GA

- Belcour's method very well suited for GA
  - compact description
  - limited and intuitive parameters
- For Real-Time rendering:
  - 2 layers + possible participating media inbetween

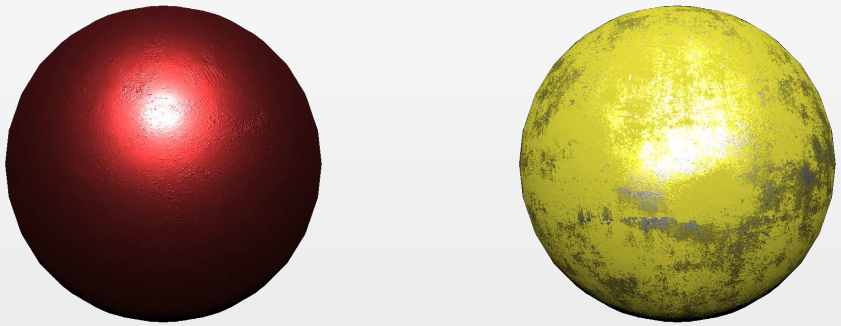
η_1	refractive index of the top layer
η_2	refractive index of the bottom layer
α_1	roughness of the top layer
α_2	roughness of the bottom layer
h	depth of the partecipating media
σ_s^R, σ_s^G , σ_s^B	scattering cross section values
σ_a^R, σ_a^G , σ_a^B	absorption values
g	anisotropic factor

If no participating media, only these 4 control final appearance

If a texture is applied to one layer, then these are used as weights for the texel value

Proposed GA: rules & constraints

- Pre-experimentation stage
  - Constraints and rules depending on the *family* of the material
    - *roughcoat*
      - smooth bottom layer + rough top layer
      - high roughness in the top layer, lower one in the bottom
    - *clearcoat*
      - roughness value of the top layer is lower than the value of the bottom
    - Further differences if bottom layer is a *conductor* (like e.g., metal) or a *dieletric* (like e.g., plastic) material

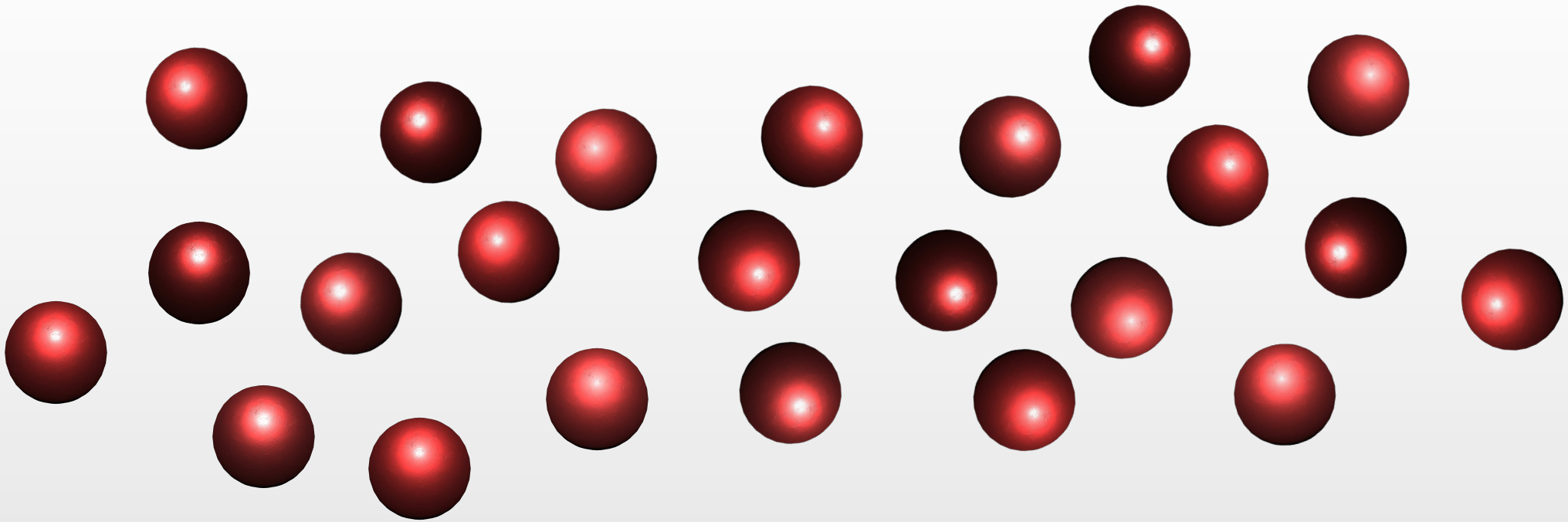


Proposed GA: rules & constraints

	Bottom Layer: conductor	**Bottom Layer: dielectric**	**Layer with textured roughness**
roughcoat	$\alpha_1 \geq 0.1$ $\alpha_2 < 0.1$ $1.0 < \eta_1 \leq 2.0$ $\lvert\eta_1 - \eta_2\rvert \in (0.0, 1.5]^{\text{a}}$	$\alpha_1 \geq 0.1$ $\alpha_2 < 0.01$ $1.0 < \eta_1 \leq 2.0$ $\lvert\eta_1 - \eta_2\rvert \in (0.0, 0.5]^{\text{b}}$ $\eta_2 \geq 1.0$	top layer
clearcoat	$\alpha_1 < 0.1$ $\alpha_2 \geq 0.1$ $1.0 < \eta_1 \leq 2.0$ $\lvert\eta_1 - \eta_2\rvert \in (0.0, 1.5]^{\text{a}}$	$\alpha_1 < 0.01$ $\alpha_2 \geq 0.01$ $1.0 < \eta_1 \leq 3.0$ $\lvert\eta_1 - \eta_2\rvert \in (0.0, 3.0]^{\text{b}}$ $\eta_2 \geq 1.0$	bottom layer
roughcoat/ clearcoat with participating media	$h \in [0.05, 40.0]^{\text{c}}$ $\sigma_s^R, \sigma_s^G, \sigma_s^B \in [0.0, 1.0]$ $\sigma_a^R, \sigma_a^G, \sigma_a^B \in [0.0, 1.0]$ $\eta_2 < 1.0^{\text{d}}$	$h \in [0.05, 40.0]^{\text{c}}$ $\sigma_s^R, \sigma_s^G, \sigma_s^B \in [0.0, 1.0]$ $\sigma_a^R, \sigma_a^G, \sigma_a^B \in [0.0, 1.0]$	—

Proposed GA: initialization

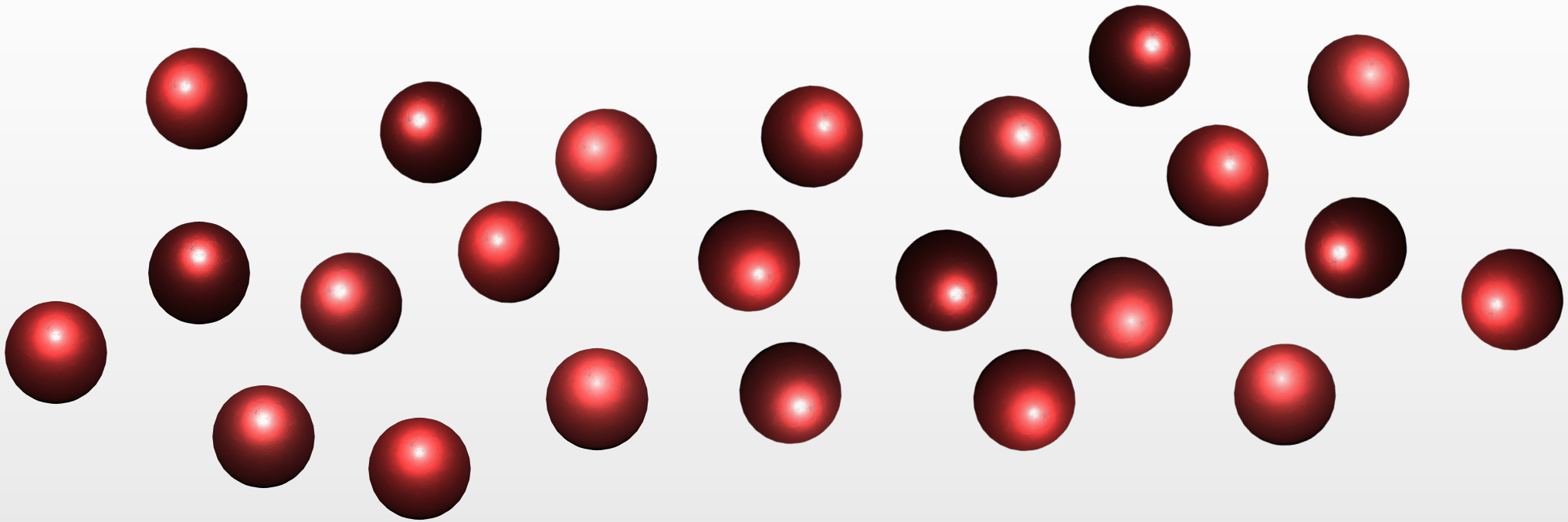
- Given a target material
  - N individuals of the same family are created
    - N ∈ [50,100]
    - chromosomes generated randomly
    - but values must follow the rules & constraints



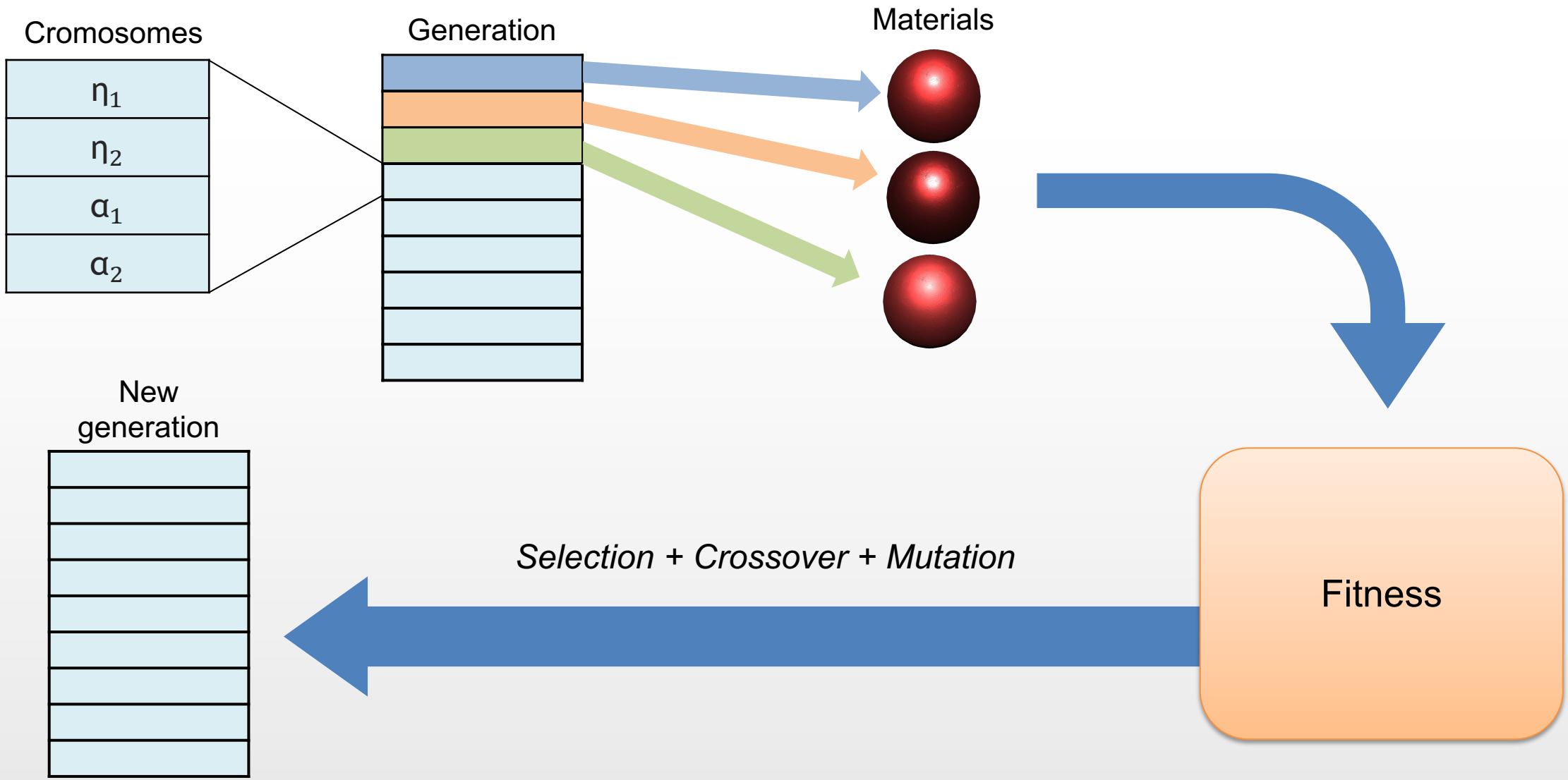
Proposed GA: initialization

- Given a target material
  - N individuals of the same family are created
    - N ∈ [50,100]
    - chromosomes generated randomly
    - but values must follow the rules & constraints

Each individual is then ranked using a fitness function
(coming soon…)



Proposed GA


Cromosomes
η_1
η_2
α_1
α_2
Generation
Materials
New generation
Selection + Crossover + Mutation
Fitness

Proposed GA

Selection

- tournament selection
- p=1
  - = selection of only the best individuals
- size = 4.
  - = high level of variety in the selected materials

Crossover

- each couple has p=0.9 to generate offspring
  - to further enhance the variety
- Uniform crossover
  - each chromosome has p= 0.25 of being swapped
  - highly improbable to have crossover of all the values at the same time
  - **2-step process to consider rules & constraints**

Mutation

- can occur with p= 0.4
- If activated, each chromosome has p=0.1 to mutate
- a new value is created randomly, but respecting rules & constraints

Proposed GA: termination

- Termination = fixed number of generations

Proposed GA: fitness function

- to measure in a simple but effective way the *difference* between the target and generated genotypes

  - Chebyshev distance:

$$d_{ch}(g, tm) = \max_i \{|g[i] - tm[i]|\}$$

  - Euclidean distance:

$$d_{eu}(g, tm) = \sqrt{\sum_i (g[i] - tm[i])^2}$$

  - Manhattan distance:

$$d_{ma}(g, tm) = \sum_i |g[i] - tm[i]|$$

g is an individual

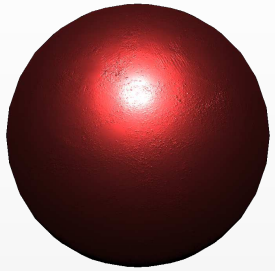
tm is the target material

i represents each of the chromosomes

Experimental evaluation

- Simple test scene
- 3 target materials

Validation metal (clearcoat)



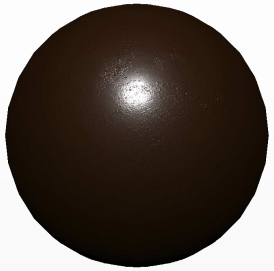
η_1: 1.2
η_2: 0.8
α_1: 0.03
α_2: 0.1

Validation painted metal (roughcoat)

η_1: 1.2
η_2: 0.8
α_1: 2.5*
α_2: 0.099

*roughness values from a texture

Validation metal depth (clearcoat with participating media)



η_1: 1.460
η_2: 0.8
α_1: 0.05
α_2: 0.45
σ_s^R: 0.1, σ_s^G: 1.0 , σ_s^B: 0.0
σ_a^R: 0.0, σ_a^G: 1.0 , σ_a^B: 0.0
g: 0.8

The texture is used as base color

Experimental evaluation

For each combination of
target material & fitness function

36 runs of the GA
population 50
termination 5

For each run
- we have considered the material with the best fitness
- i.e., with minimum distance to the target material

Experimental evaluation

For each combination of target material & fitness function

36 runs of the GA
population 50
termination 5

For each run
- we have considered the material with the best fitness
- i.e., with minimum distance to the target material

CIEDE2000 ΔE_0^*0

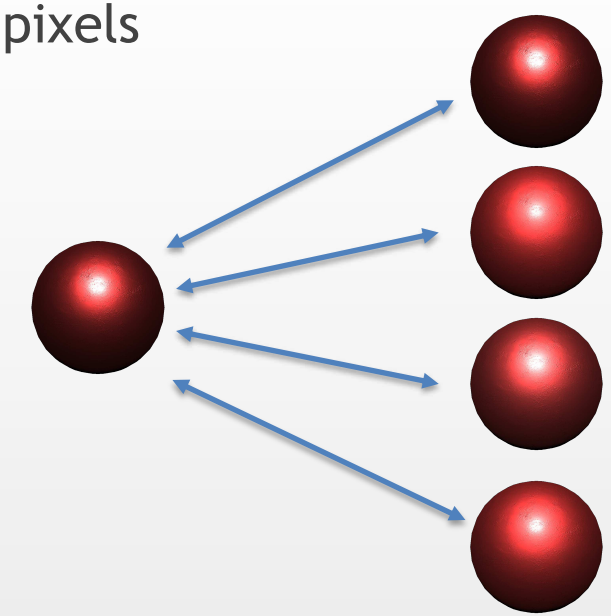
Experimental evaluation

- CIEDE2000 ΔE_0^*0
  - Perceptual measure between 2 colors

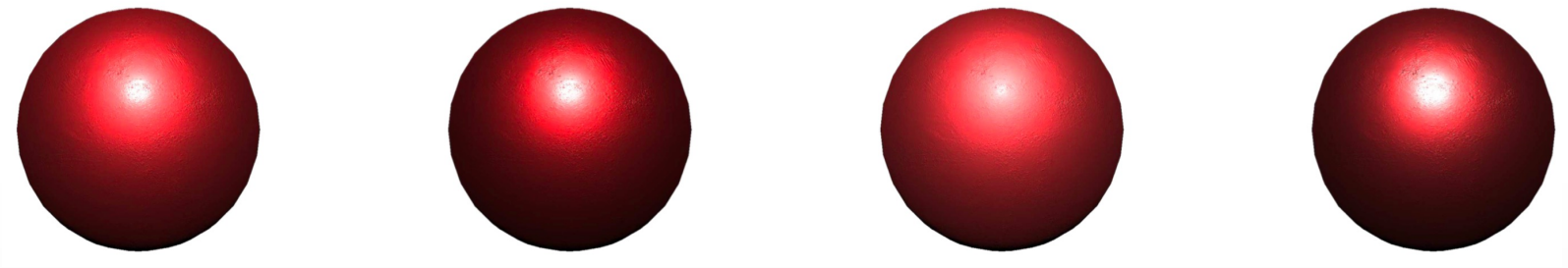
$\Delta E_0^*0 \leq 1.0$	color difference is not perceptible by human eyes
$\Delta E_0^*0 \in [1.0,2.0]$	color difference is perceptible through close observation
$\Delta E_0^*0 \in [2.0,10.0]$	color difference is perceptible at a glance
$\Delta E_0^*0 \in [10.0,49.0]$	colors are more similar than opposite
$\Delta E_0^*0 \in [49.0,100.0]$	colors are exact opposite

Experimental evaluation

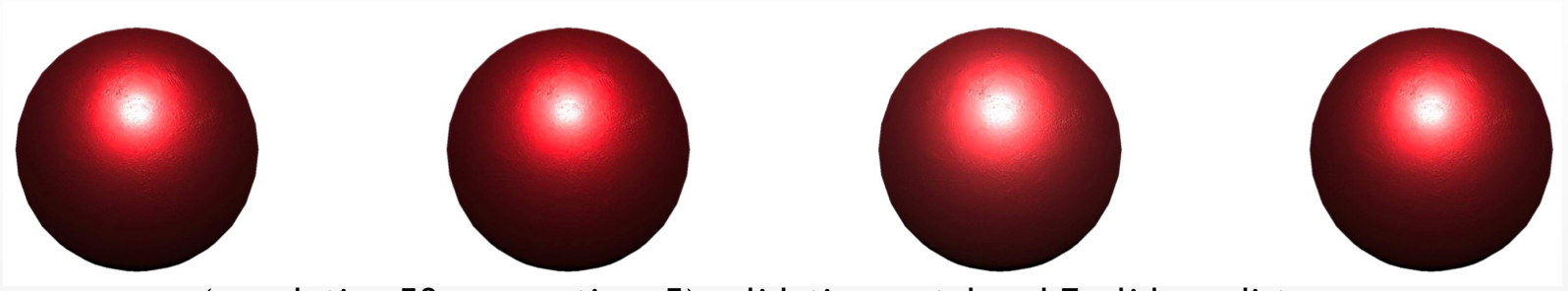
- ΔE_0^*0 applied to:
  - each pixel of the image rendered using the target material
  - and the corresponding pixels in each of the images created using the generated materials parameters
  - mean ΔE_0^*0 value by averaging on the single pixels



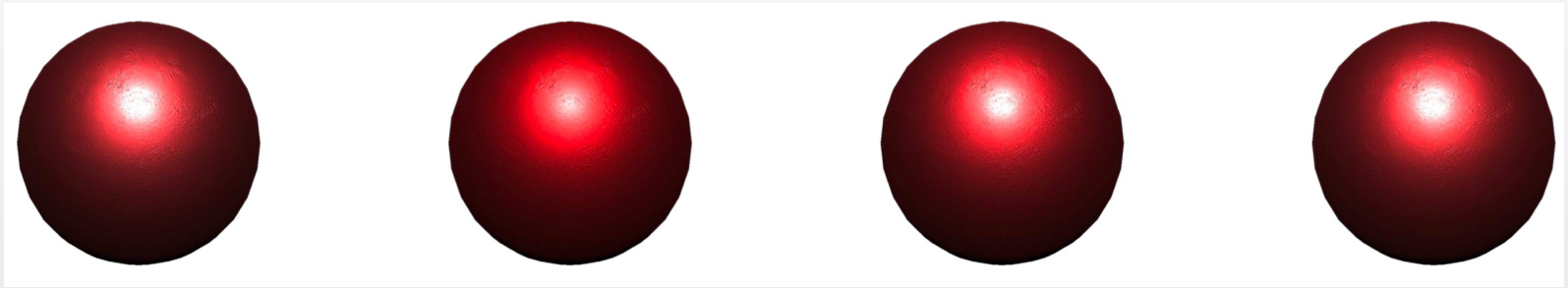
Experimental evaluation



(population 50, generations 5) validation metal and Chebyshev distance



(population 50, generations 5) validation metal and Euclidean distance



(population 50, generations 5) validation metal and Manhattan distance

Experimental evaluation

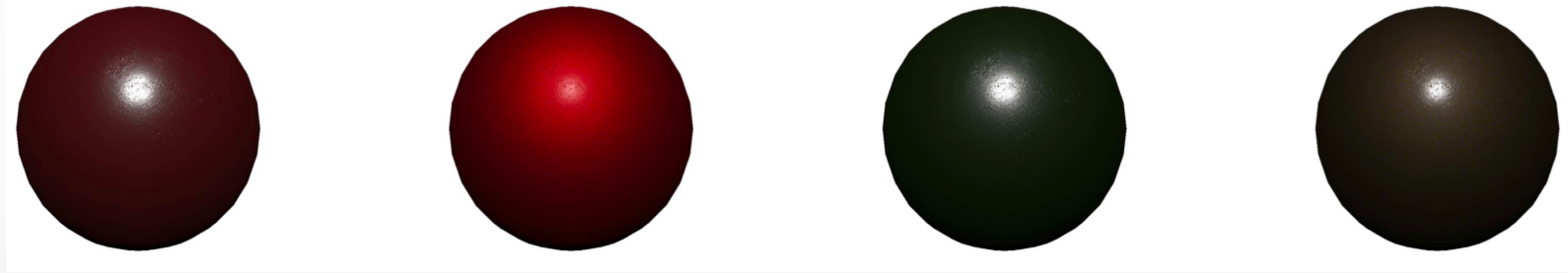
(population 50, generations 5) validation painted metal and Chebyshev distance

(population 50, generations 5) validation painted metal and Euclidean distance

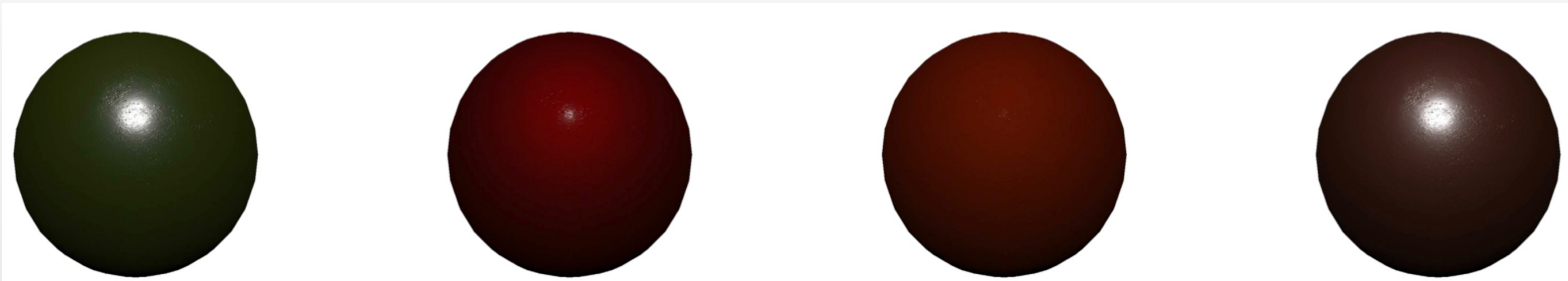
(population 50, generations 5) validation painted metal and Manhattan distance

Experimental evaluation

(population 50, generations 5) validation metal depth and Chebyshev distance



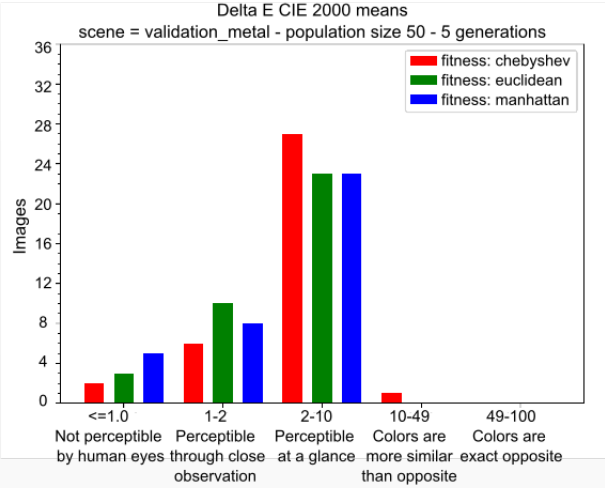
(population 50, generations 5) validation metal depth and Euclidean distance



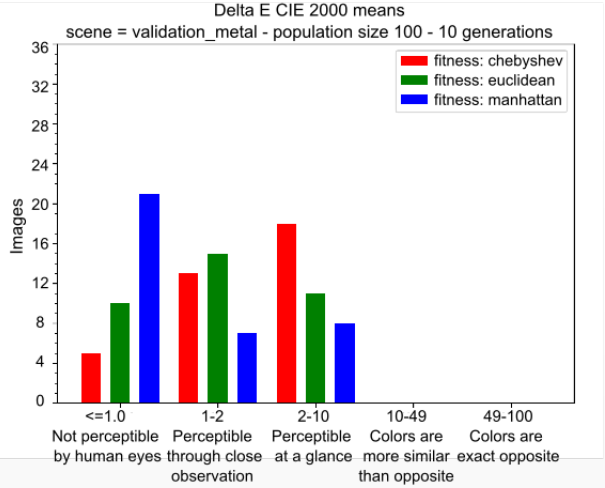
(population 50, generations 5) validation metal depth and Manhattan distance

mean ΔE_0*0 - validation metal

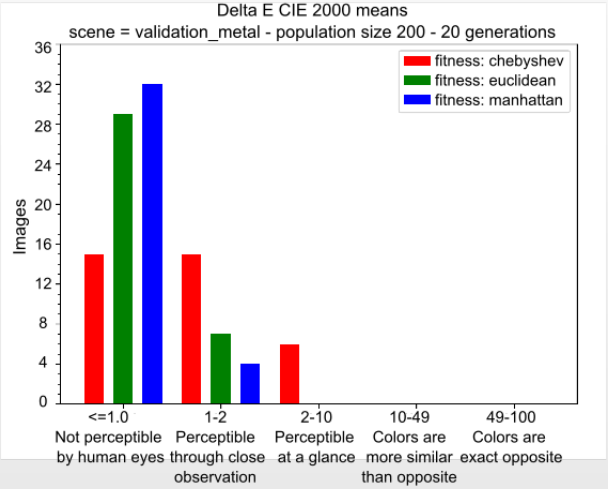
Population 50 – generations 5



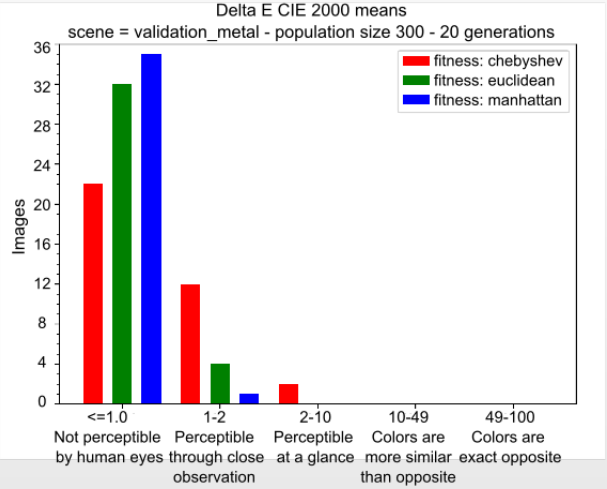
Population 100 – generations 10



Population 200 – generations 20

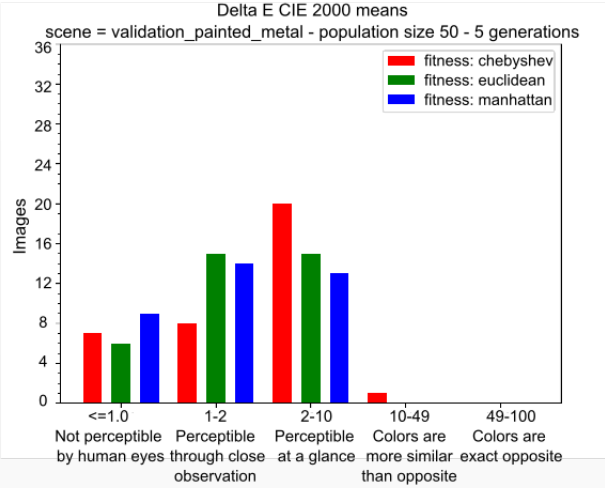


Population 300 – generations 20

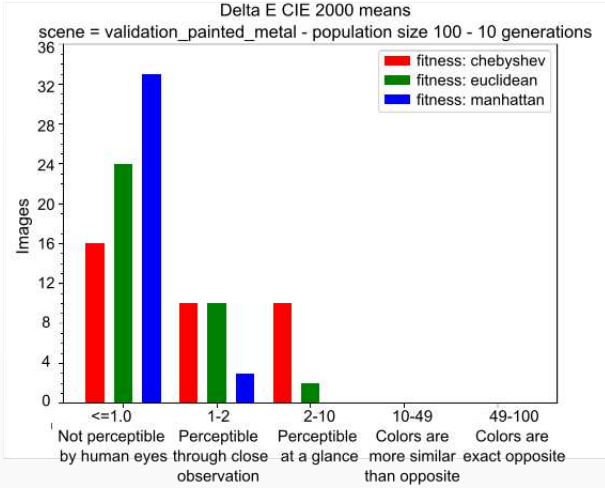


mean ΔE_0*0 - validation painted metal

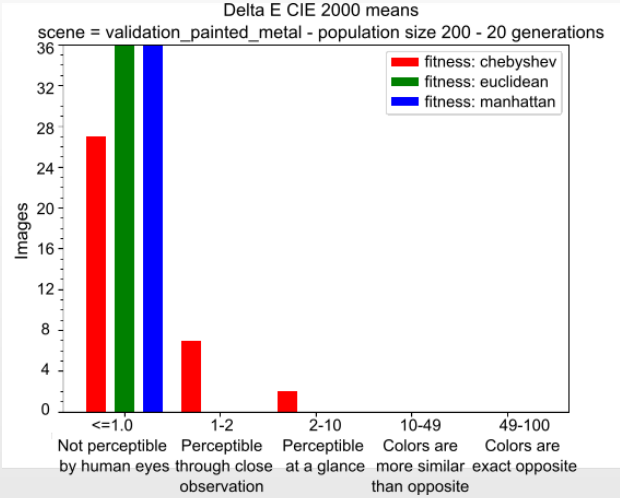
Population 50 – generations 5



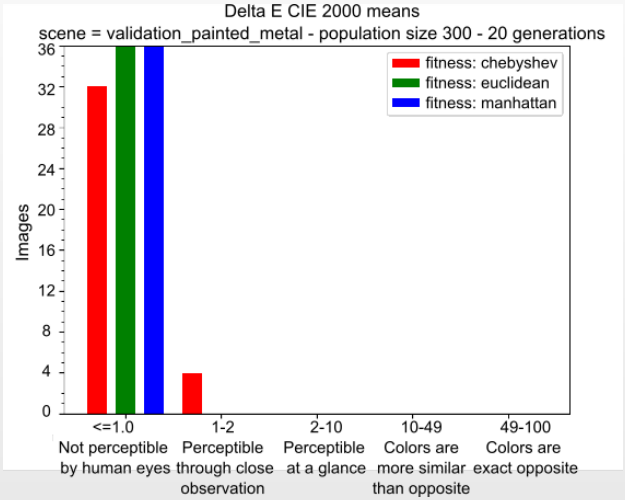
Population 100 – generations 10



Population 200 – generations 20

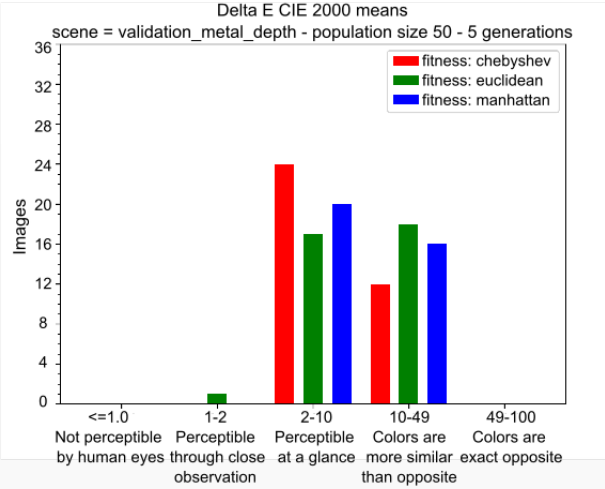


Population 300 – generations 20

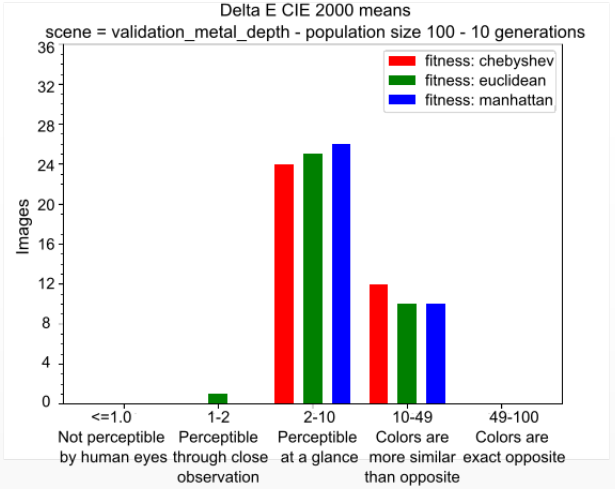


mean ΔE_0*0 - validation metal depth

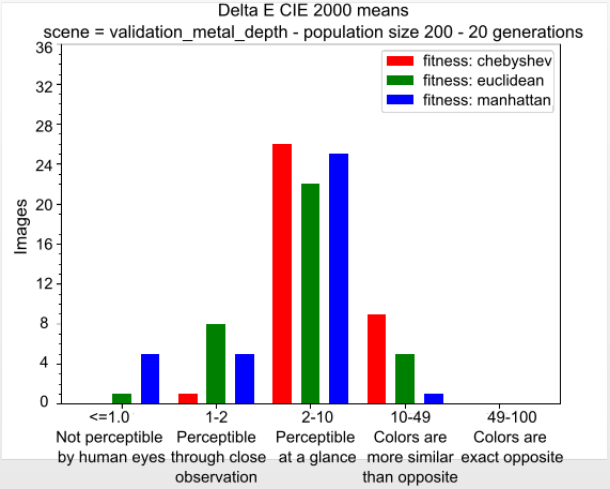
Population 50 – generations 5



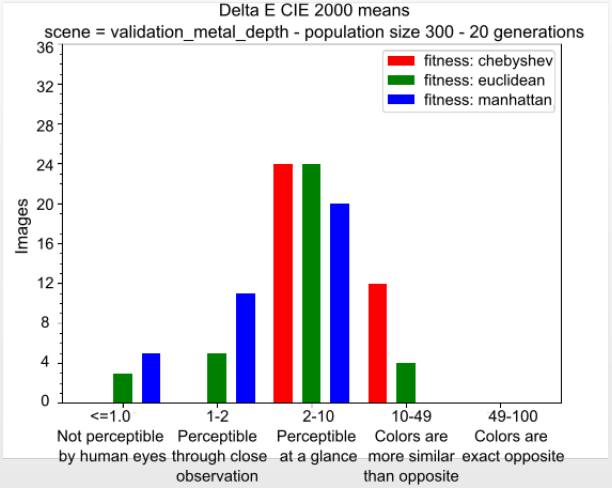
Population 100 – generations 10



Population 200 – generations 20



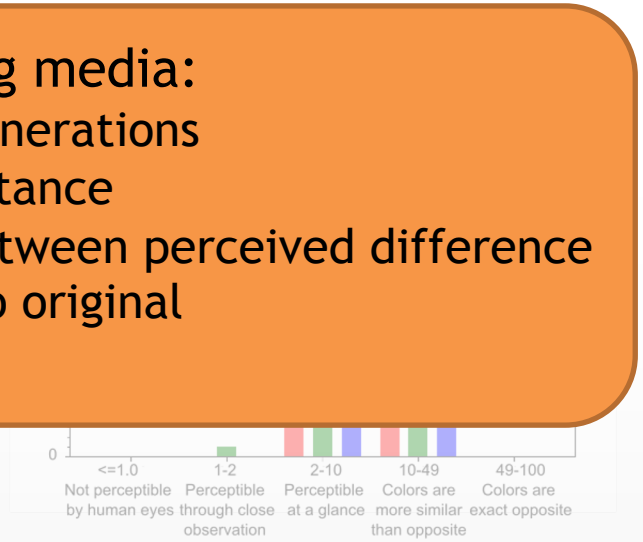
Population 300 – generations 20



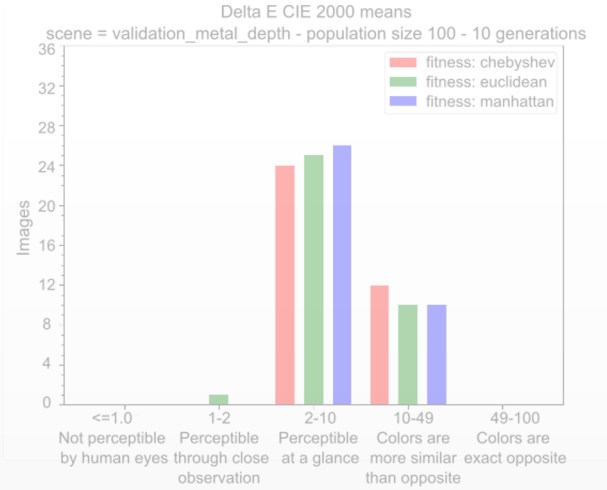
mean ΔE_0*0 - validation metal depth

With no participating media:

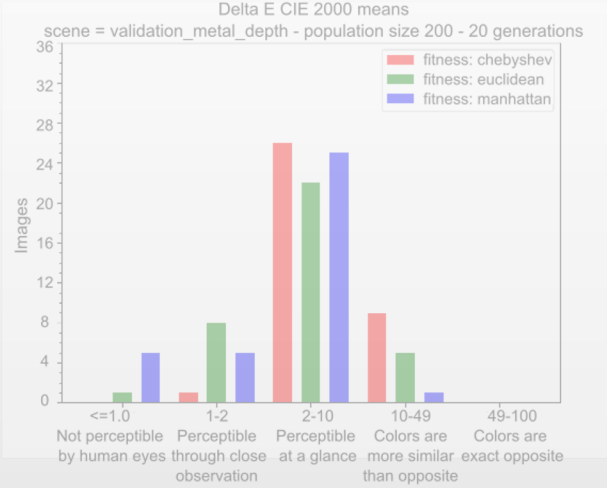
- population 50, 5 generations
- with Chebyshev distance
- → good balance between perceived difference and resemblance to original



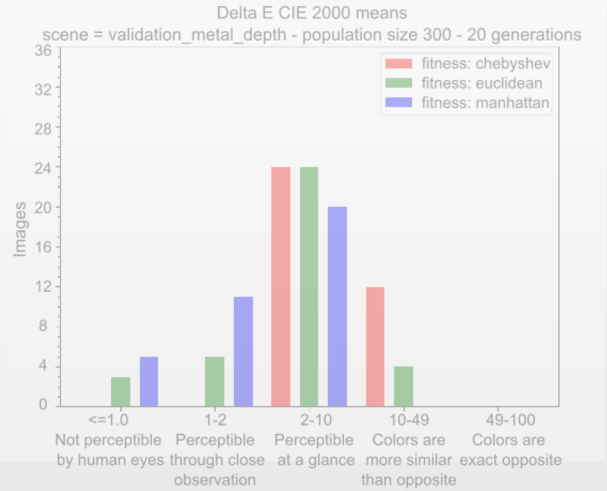
Population 100 – generations 10



Population 200 – generations 20



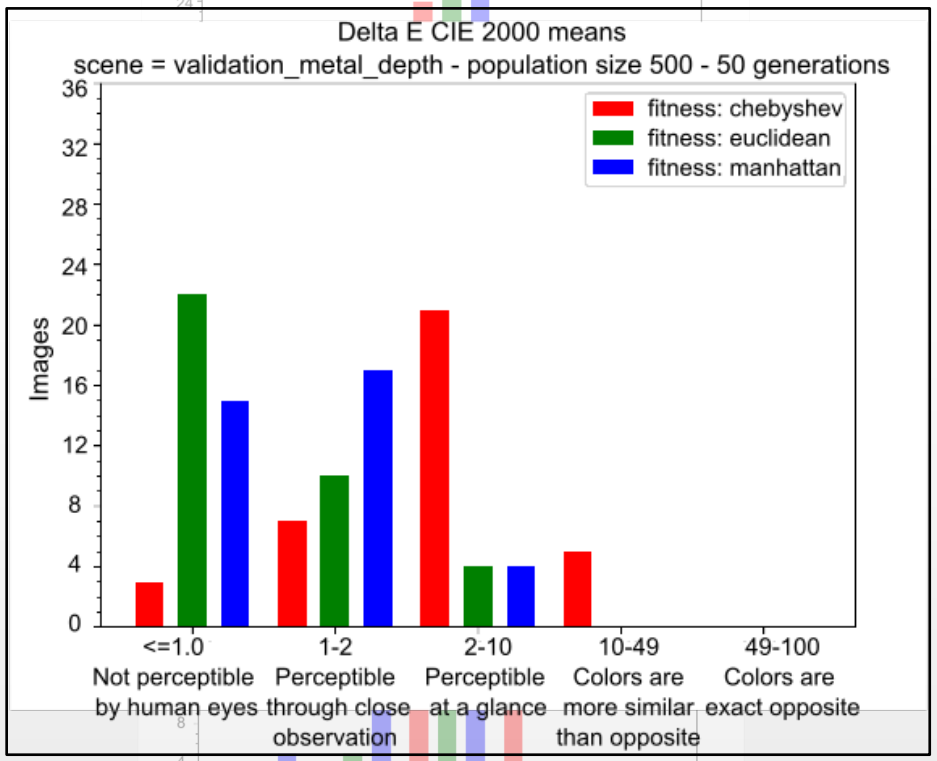
Population 300 – generations 20



mean ΔE_0*0 - validation metal depth

With no participating media:
- population 50, 5 generations
- with Chebyshev distance
- → good balance between perceived difference and resemblance to original

With participating media:
- absorption and scattering have a relevant role
- need to raise parameters value to have the same behaviour
- population 500, 50 generations

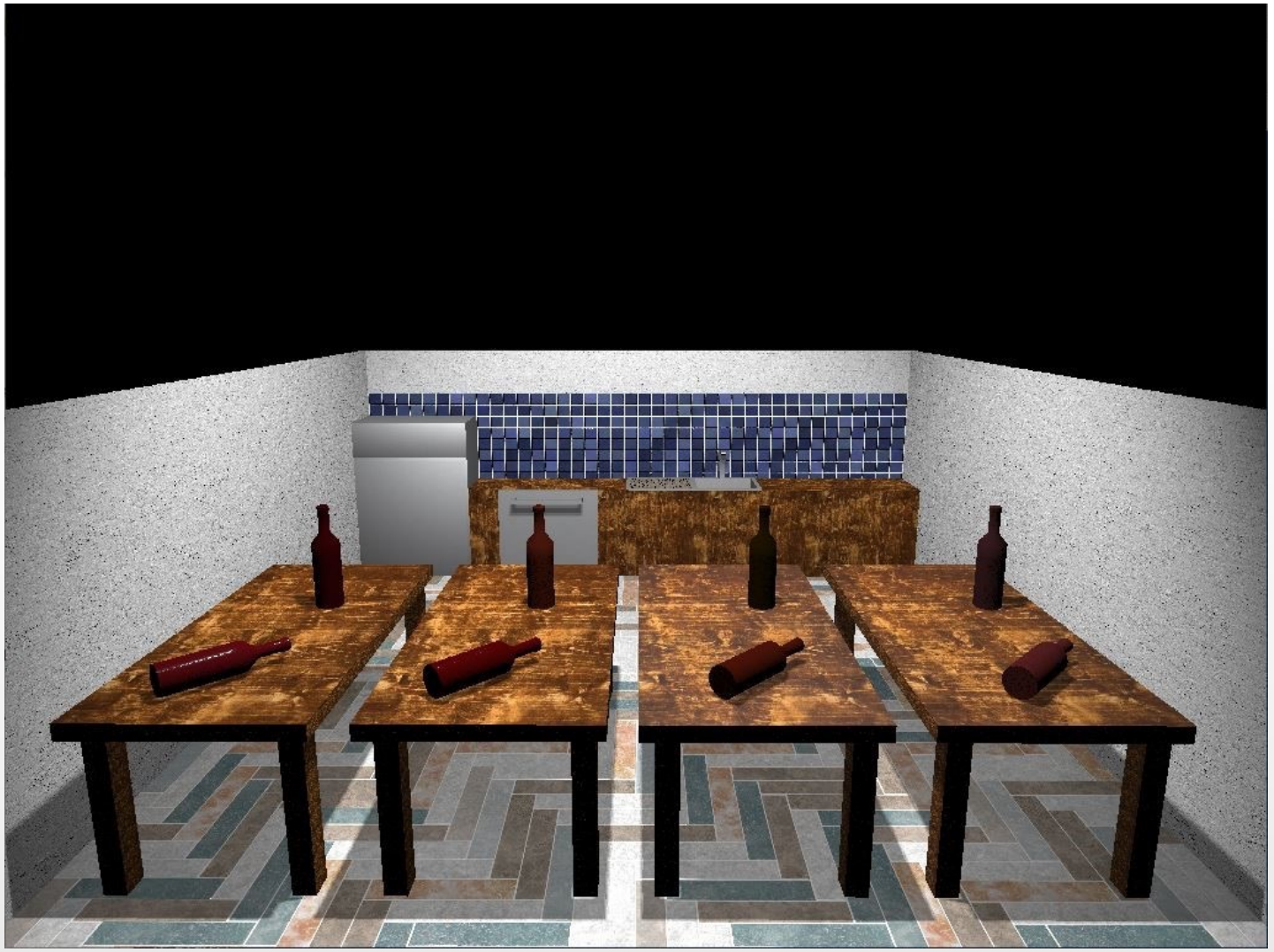


Implementation details and performance

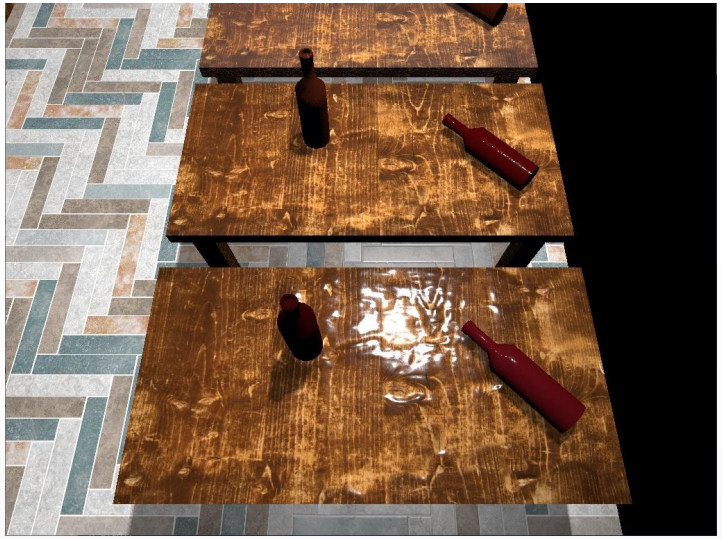
- Vulkan and C++
- GA and BRDF implementation are independent
  - GA is computed on CPU
    - Offline during scene loading, of after PCG of models
  - BRDF executed as shader on GPU
    - Using results of GA as parameters
  - On our test machine:
    - CPU Intel® Core™ i7-6700HQ, 4 cores (2.60 GHz)
    - GPU NVIDIA® GeForce® GTX 970M, 3GB GDDR5 VRAM
    - 8GB RAM DDR4 (2400 MHz)
    - GA: 30ms to generate a material with population = 100 and 10 generations
    - BRDF: real-time performances in line with Belcour's paper

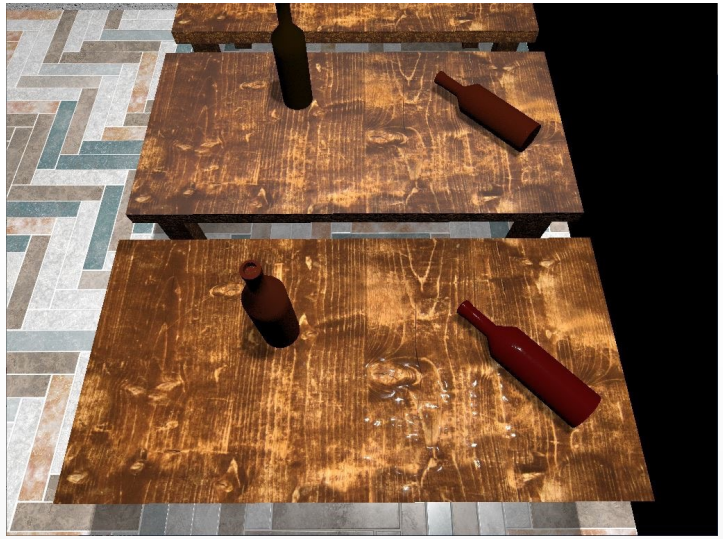
Results

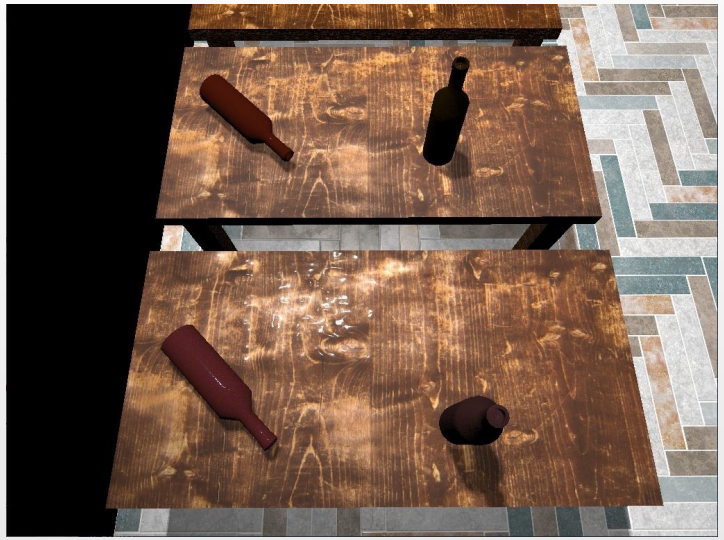
Results



Results







Results

Efficient generation of CG contents using PCG and GAs

ISSUES:

- fast dynamic update of the CG mesh
- fine control
- to be easy to manage by common PCG techniques (like GA)

Efficient generation of CG contents using PCG and GAs

ISSUES:

- fast dynamic update of the CG mesh
- fine control
- to be easy to manage by common PCG techniques (like GA)

PROPOSAL:
Parametric curves and surfaces

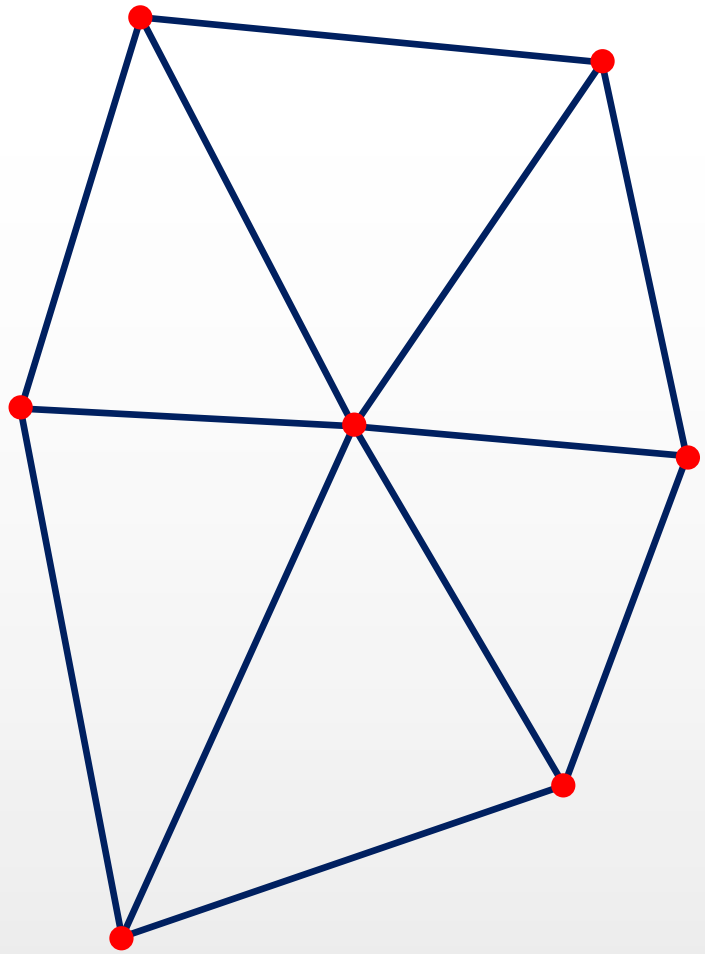
- fine control on curvatures
  (→ easy to introduce subtle modifications)
- PCG applied only on a limited set of parameters
- real-time polygonal mesh creation supported on GPU with controlled LOD

Polygon mesh: primitives

- vertices (with position)
  $V = \{v_1, v_2, \dots, v_n\}$

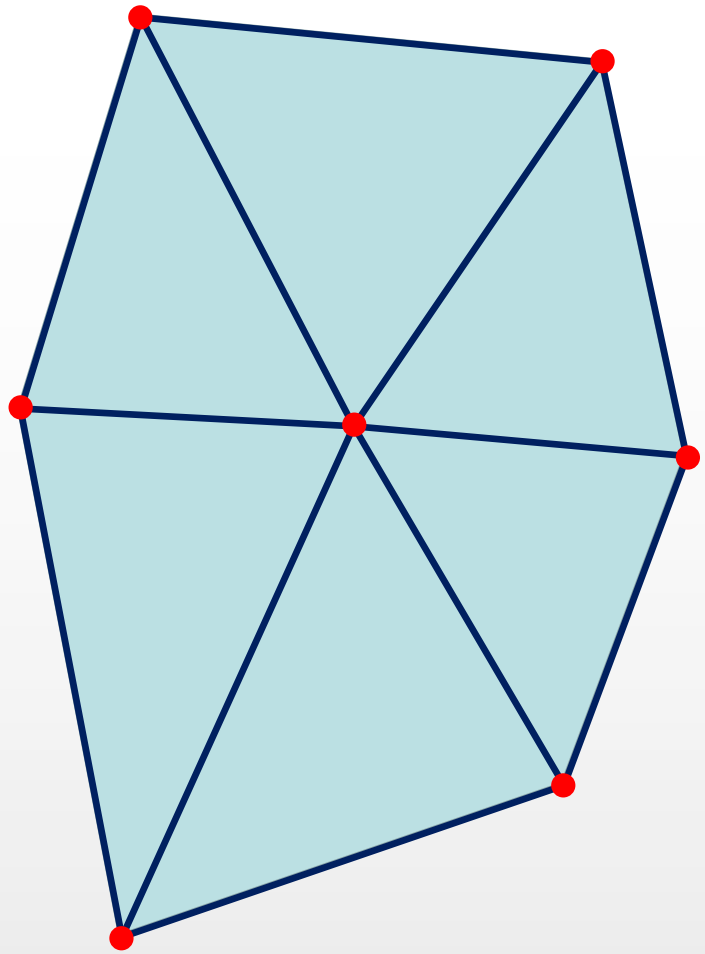
Polygon mesh: primitives

- vertices (with position)
  $V = \{v_1, v_2, \dots, v_n\}$
- edges
  $E = \{e_1, e_2, \dots, e_k\}, e_i \in V \times V$



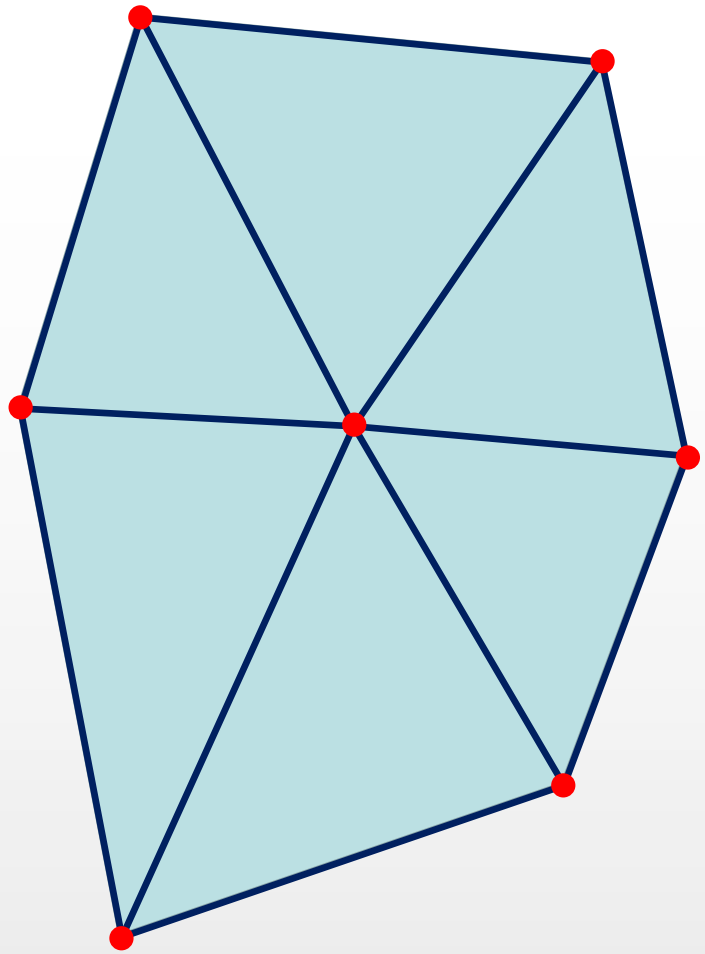
Polygon mesh: primitives

- vertices (with position)
  $V = \{v_1, v_2, \dots, v_n\}$
- edges
  $E = \{e_1, e_2, \dots, e_k\}, e_i \in V \times V$
- faces
  $\mathrm{F} = \{f_1, f_2, \dots, f_m\}\ , f_i \in V \times V \times V$



Polygon mesh: topology

- A.K.A. *connectivity*
  - information about how the vertices are connected together
  - is a **graph** in $\mathbb{R}^3$
    - vertices = nodes

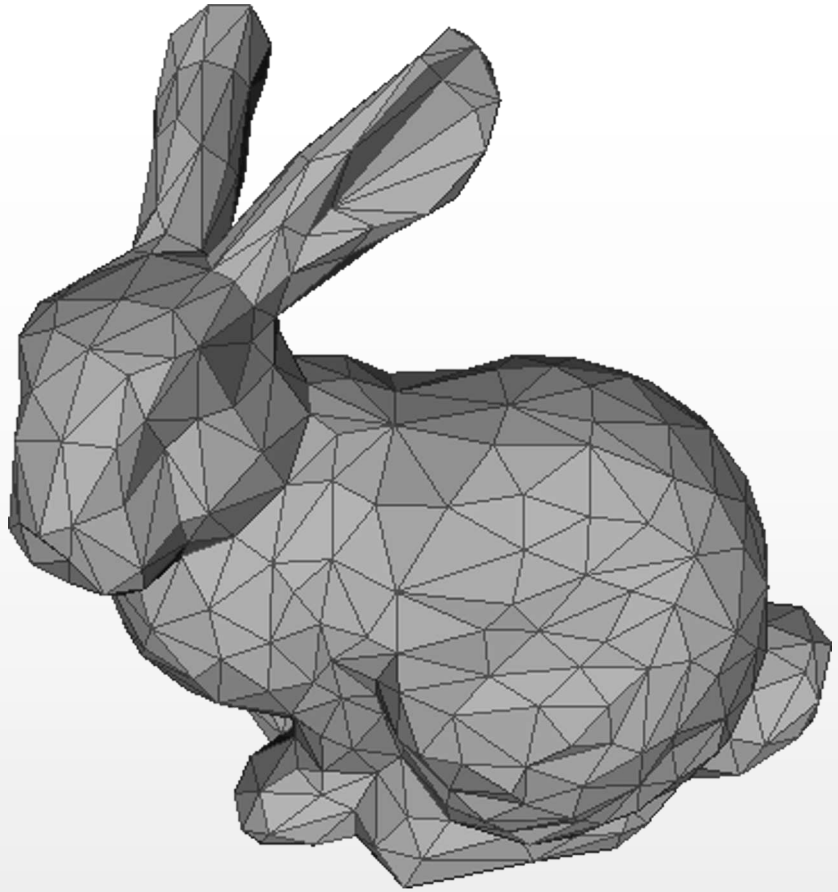


Polygon mesh

A mesh processing operation requires to analyze and (if needed) change/adapt the topology

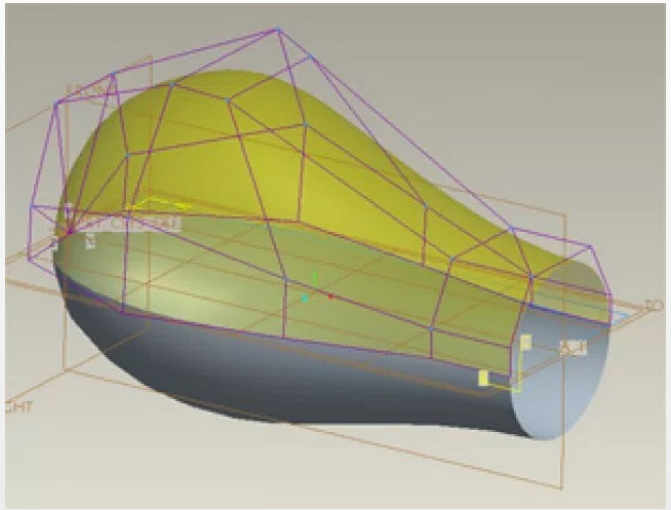
- to maintain curvature
- to re-triangulate
- etc

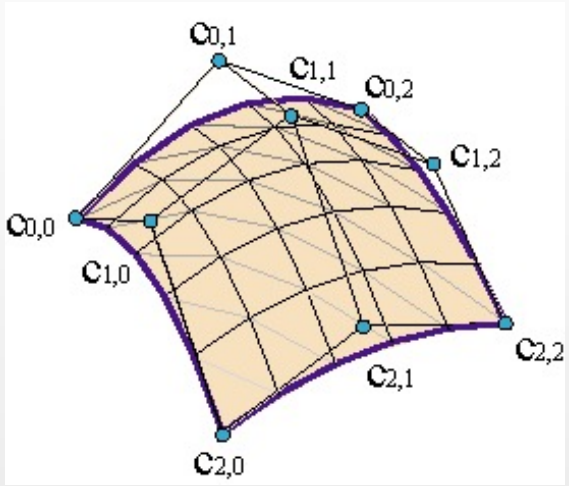
In PCG, the elaboration/correction of polygon meshes after the automatic generation/evolution is a critic step

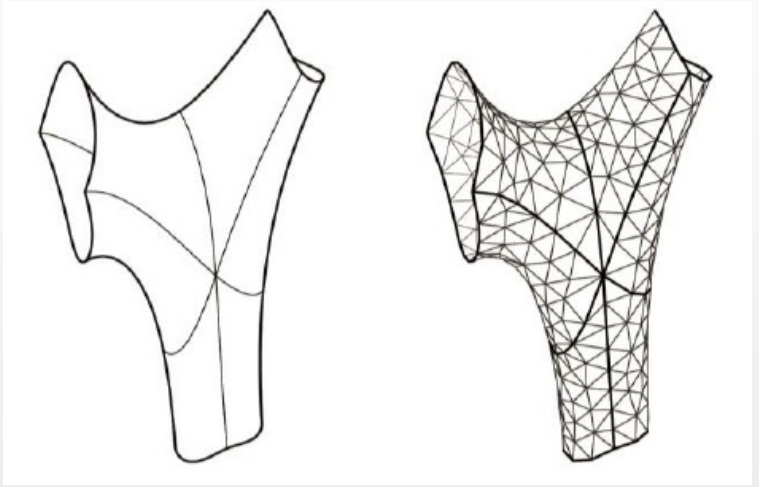


Another approach: parametric curves/surfaces

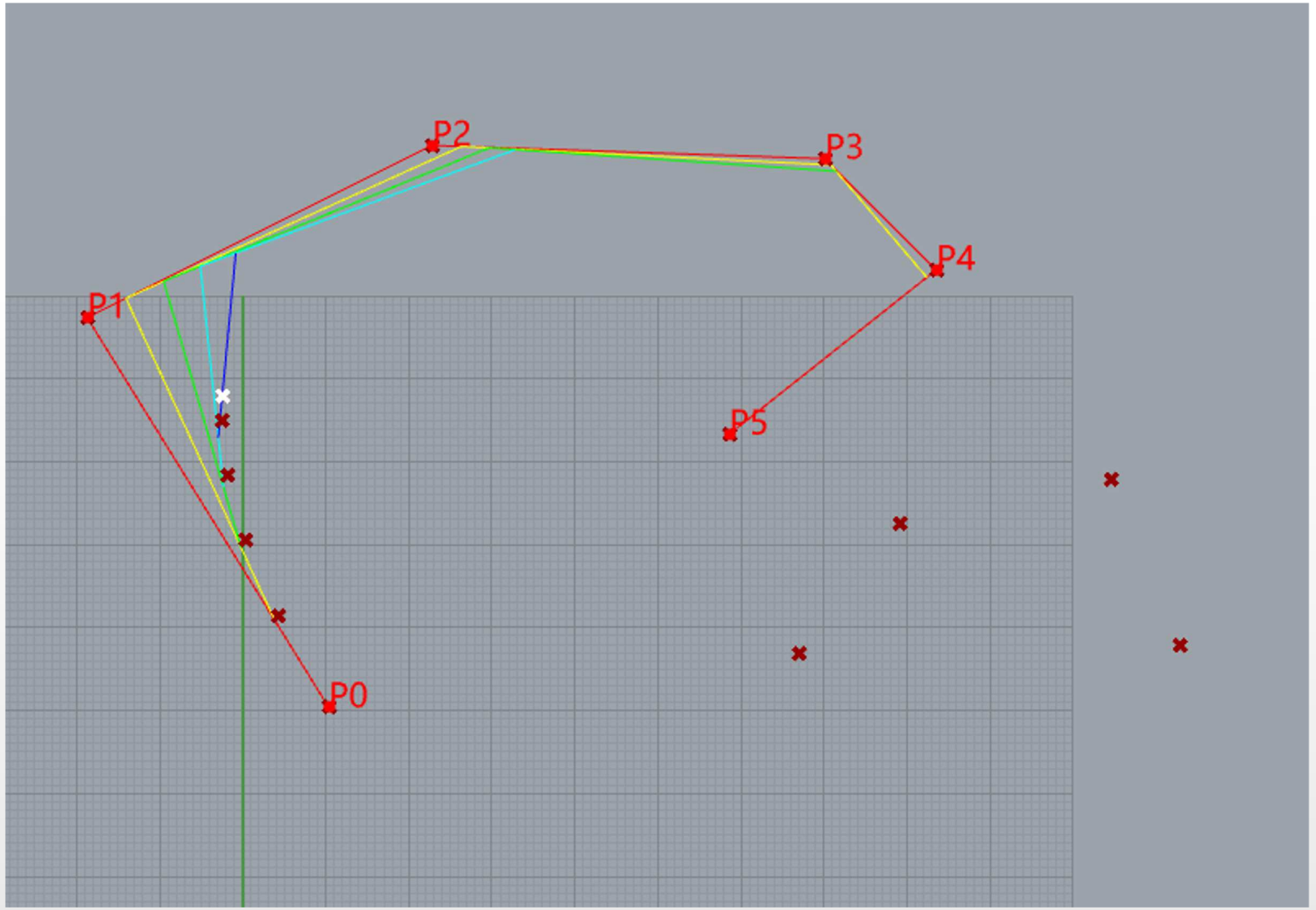
- Curves and surfaces described by a mathematical definition
  - curvature controlled by a set of control points/vectors
    - Bezier
    - NURBS
  - Polygon meshes created from the *evaluation* of the mathematical definition



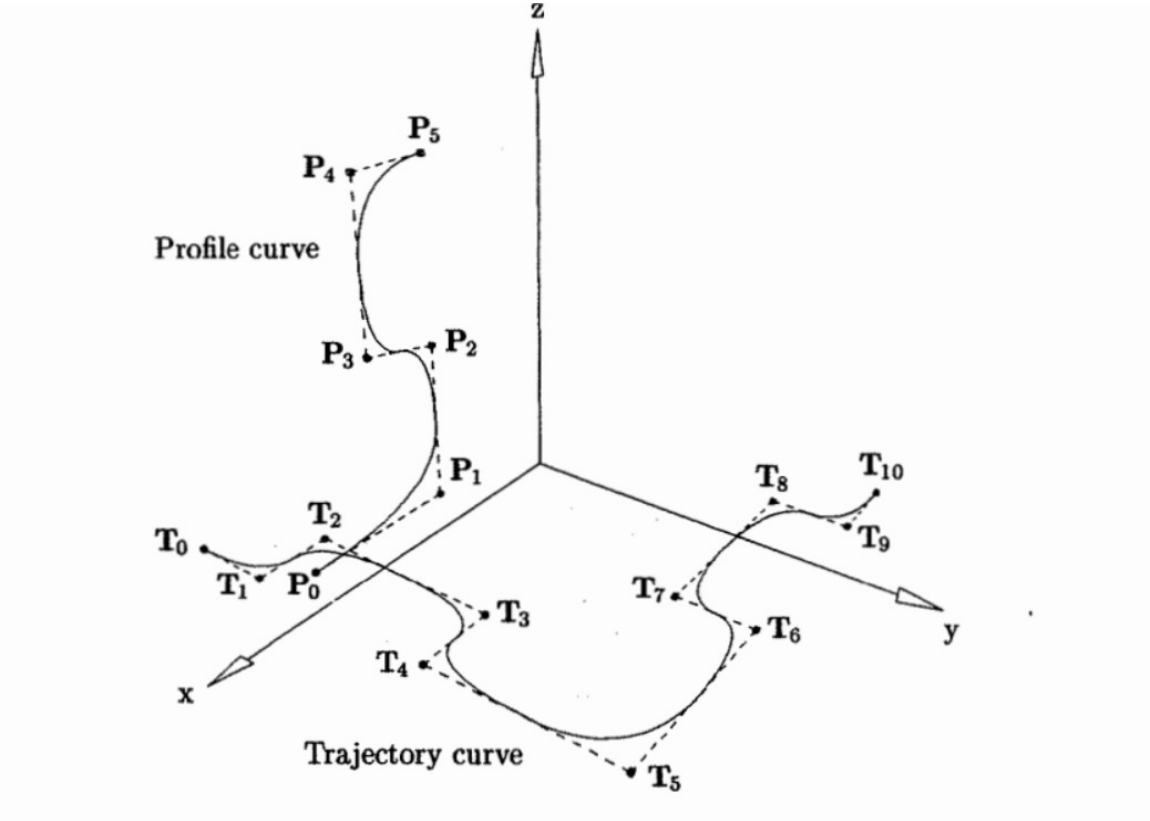


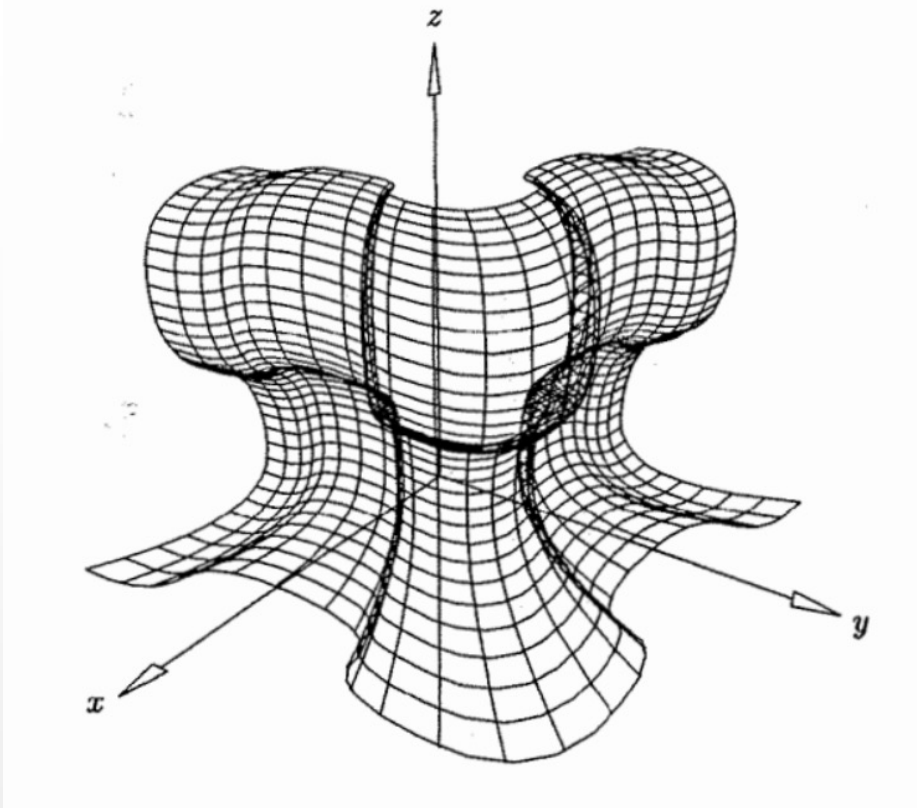


Another approach: parametric curves/surfaces

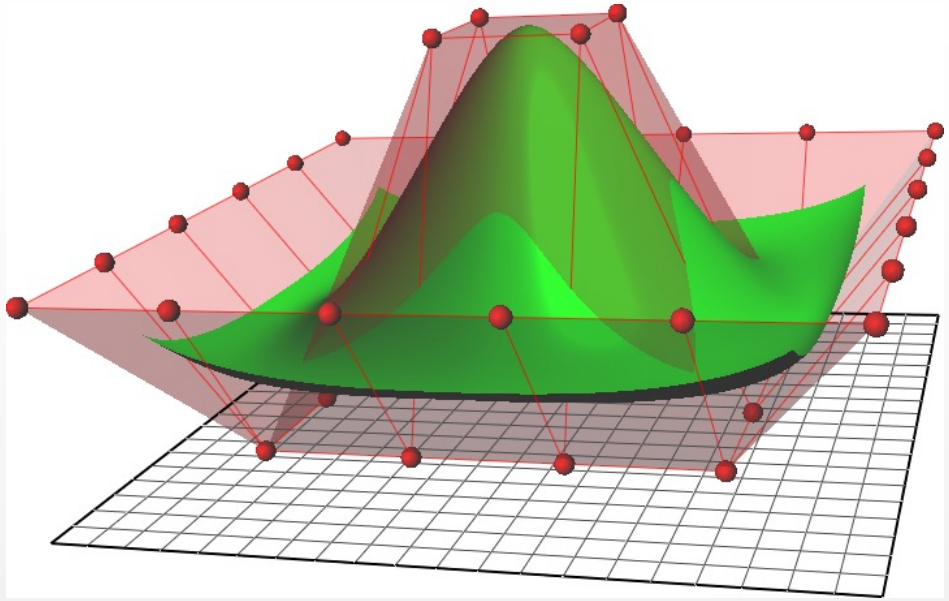


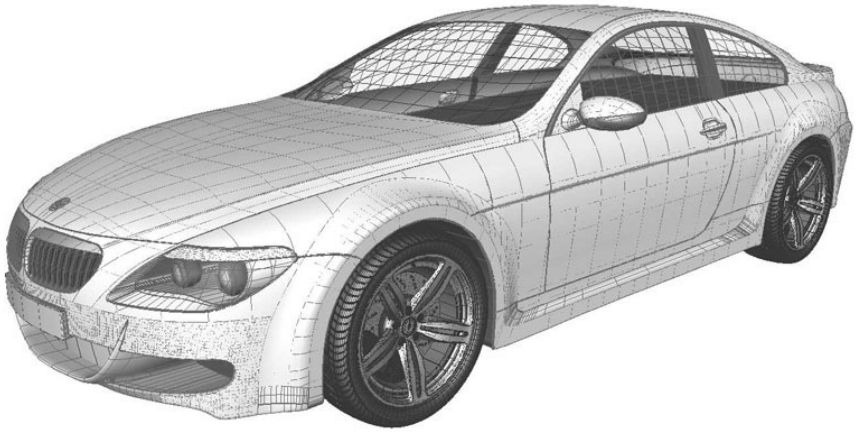
Parametric curves/surfaces

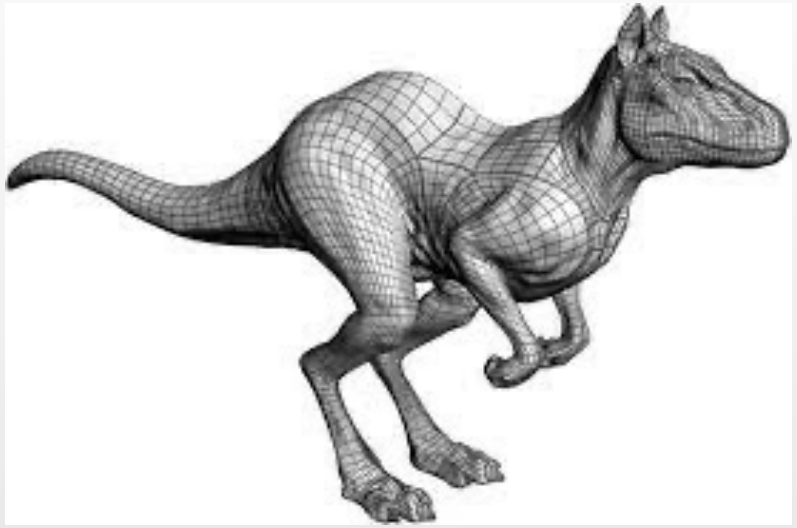




Parametric curves/surfaces

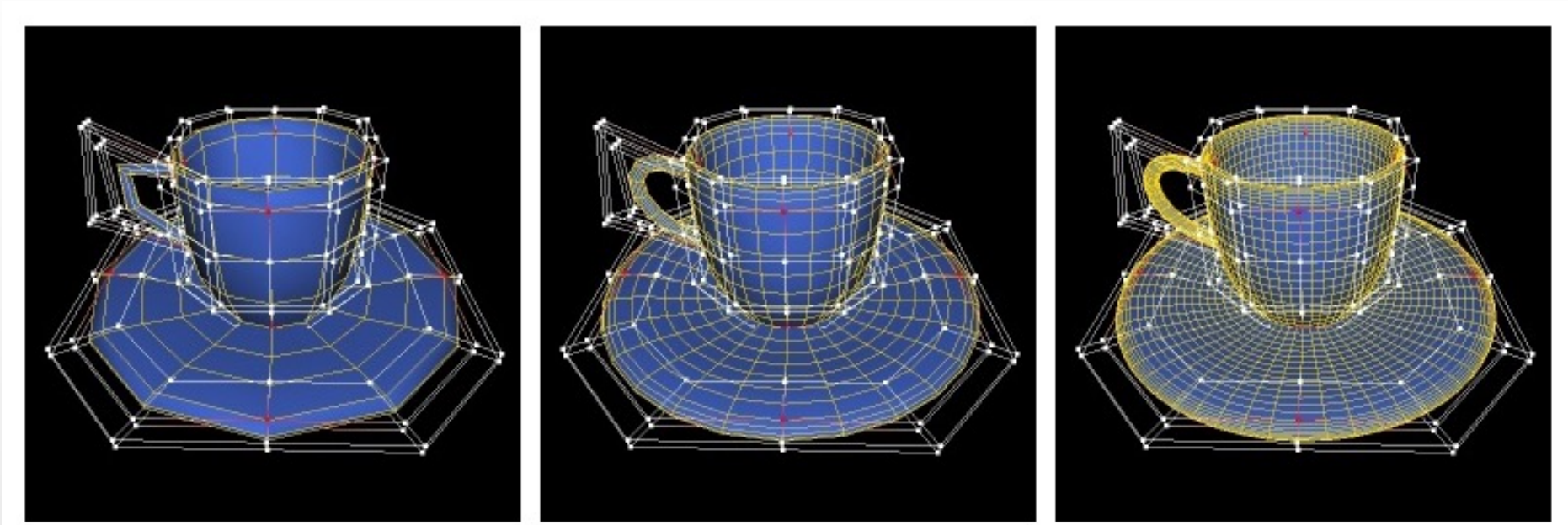






Parametric curves/surfaces

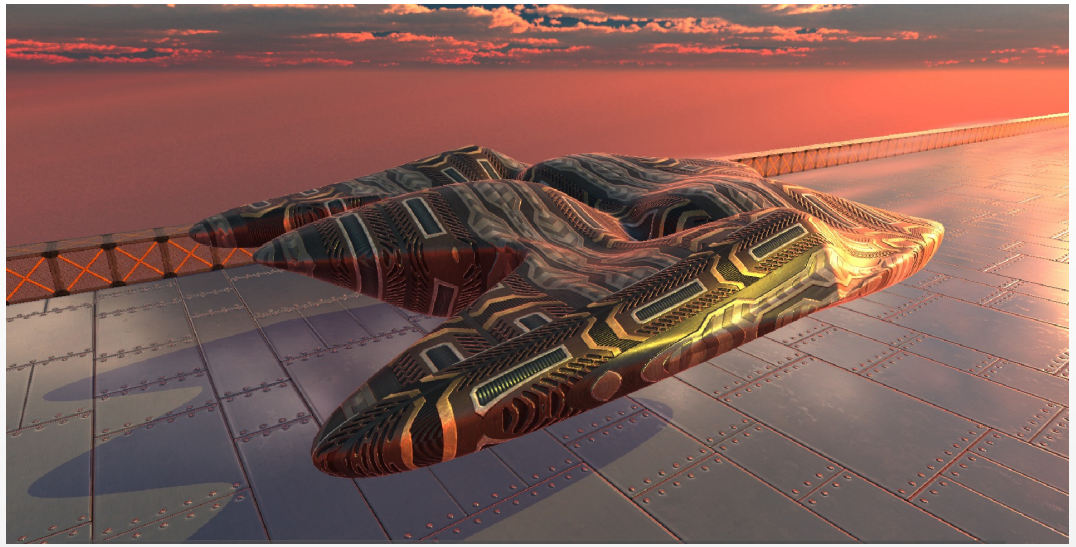
- Usually used for high-poly models with smoothed curves
- In real-time graphics:
  - Evaluation can be performed on GPU
  - allow dynamic adjustment of mesh complexity
    - Dynamic Simplification



Efficient generation of CG contents using PCG and GAs

Evolution of scifi spaceships for arcade racing game

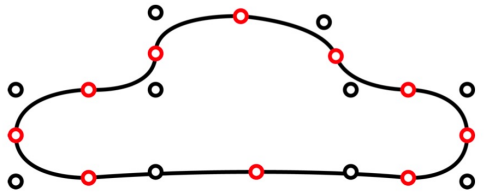
- Introduction of large variety of vehicles
- evolution based on approximate aerodynamics
- integration of preference of player about type of vehicle, track, etc, or detection of player racing style

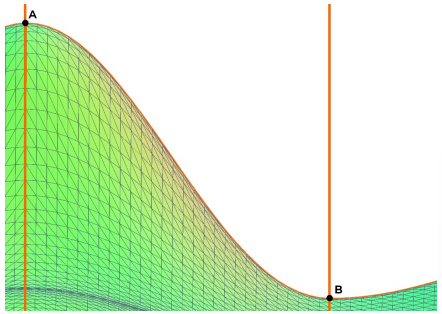


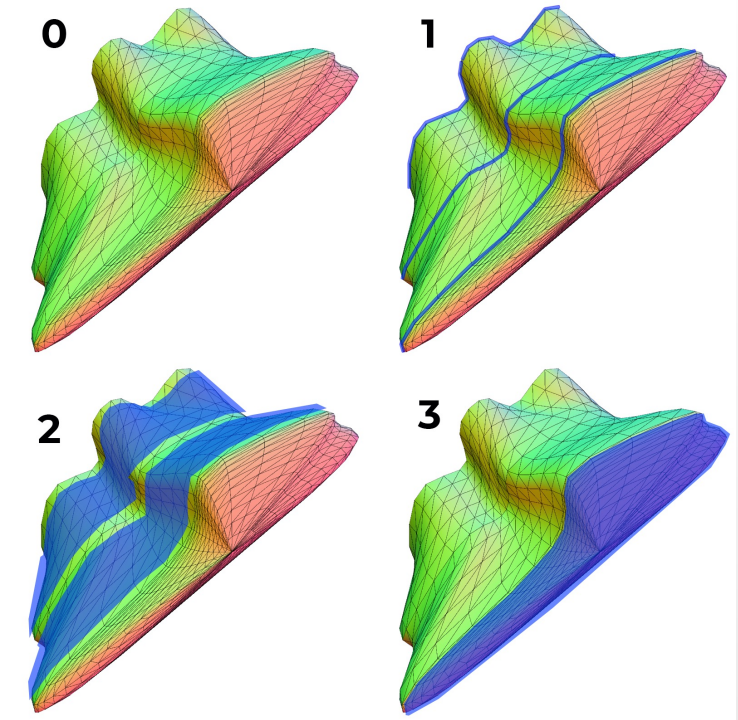
Efficient generation of CG contents using PCG and GAs

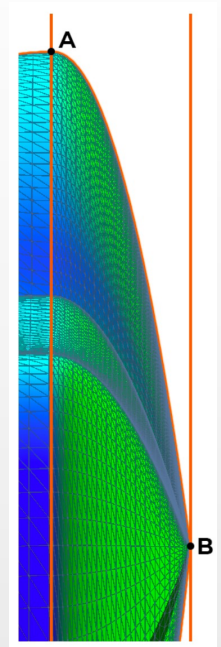
Approach: spaceship generation based on 3 sections

- 2nd order Bezier splines
- polygonal mesh built interpolating vertices between the sections
- different functions for body and end-points









Proposed GA

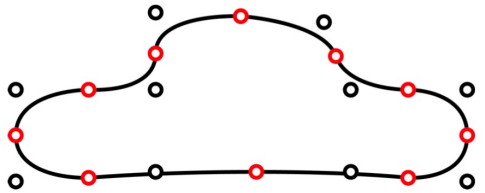
Genetic algorithm

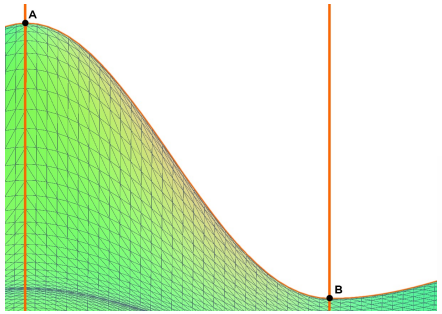
genome
3 curves
(each with 10 control points)

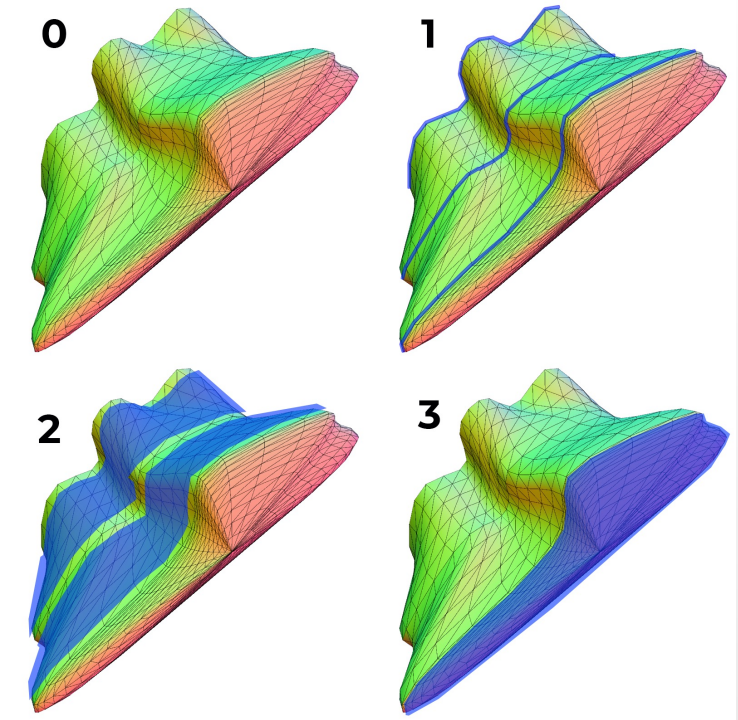
Selection
elitism + deterministic tournament

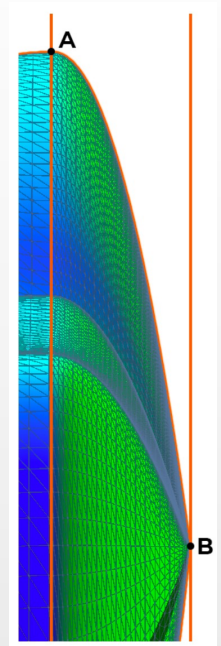
Crossover
Uniform between sections

Mutation
50% probability to mutate 5 points in a section



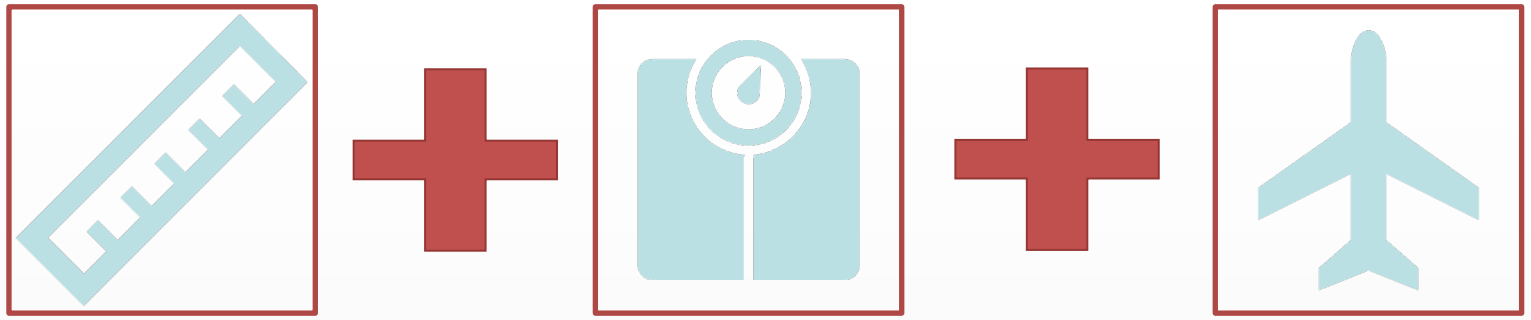






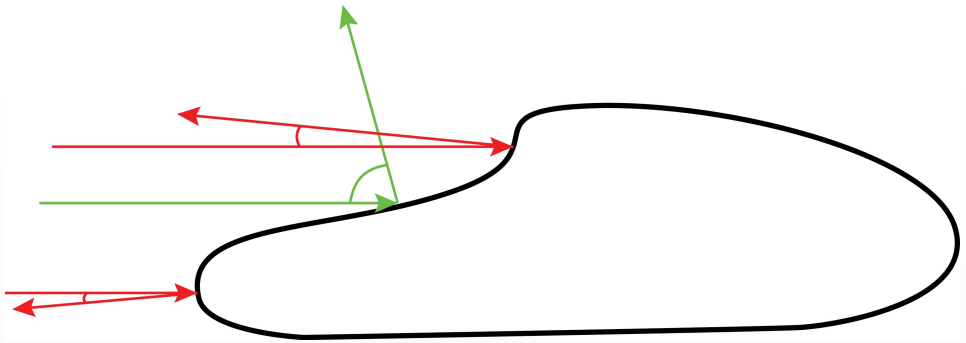
Proposed fitness function

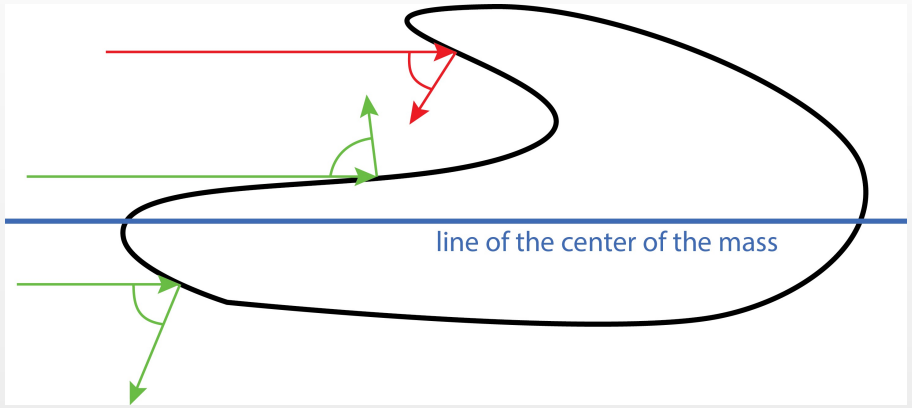
FITNESS

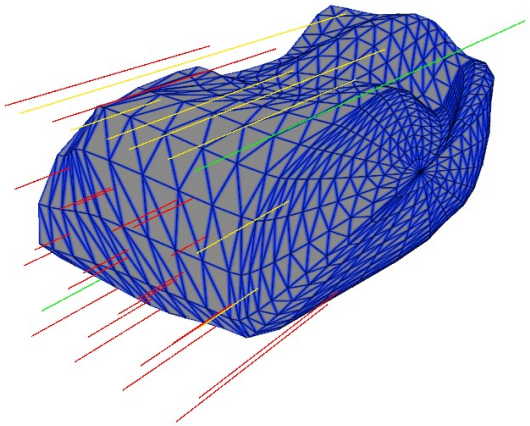


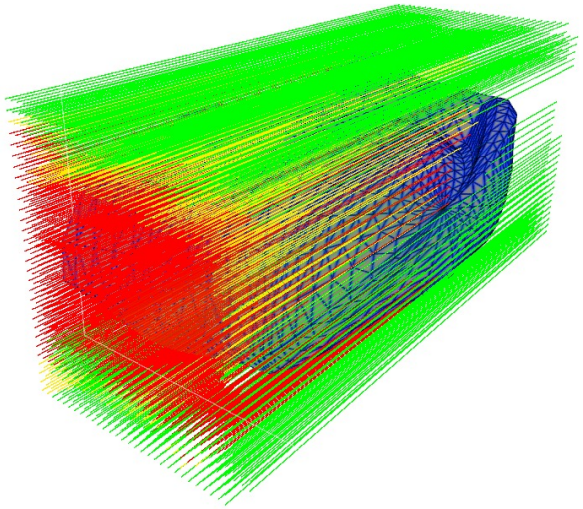
$$fitness(i) = \frac{sizeDistance(i,t)^2 + volumeDistance(i,t)^2 + aerodynamicDistance(i,t)^2}{3}$$

Proposed fitness function: approximated aerodynamics performance

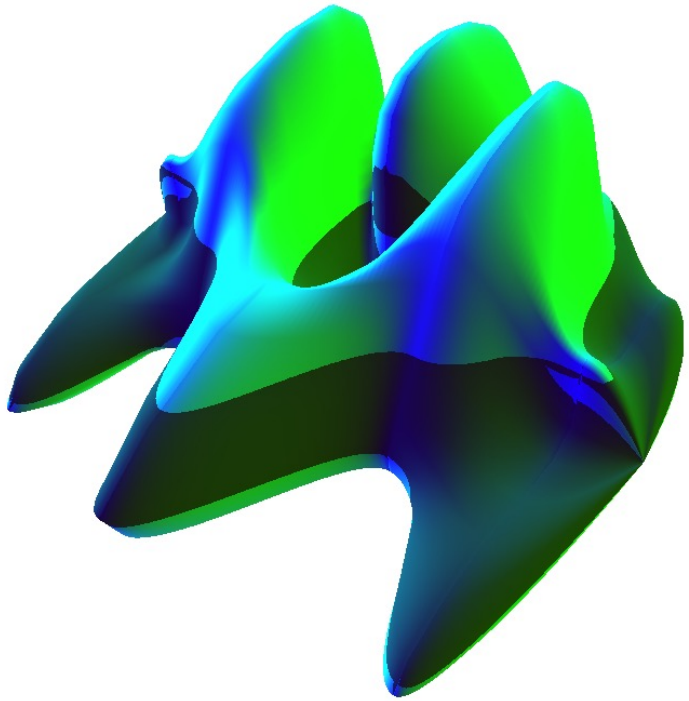


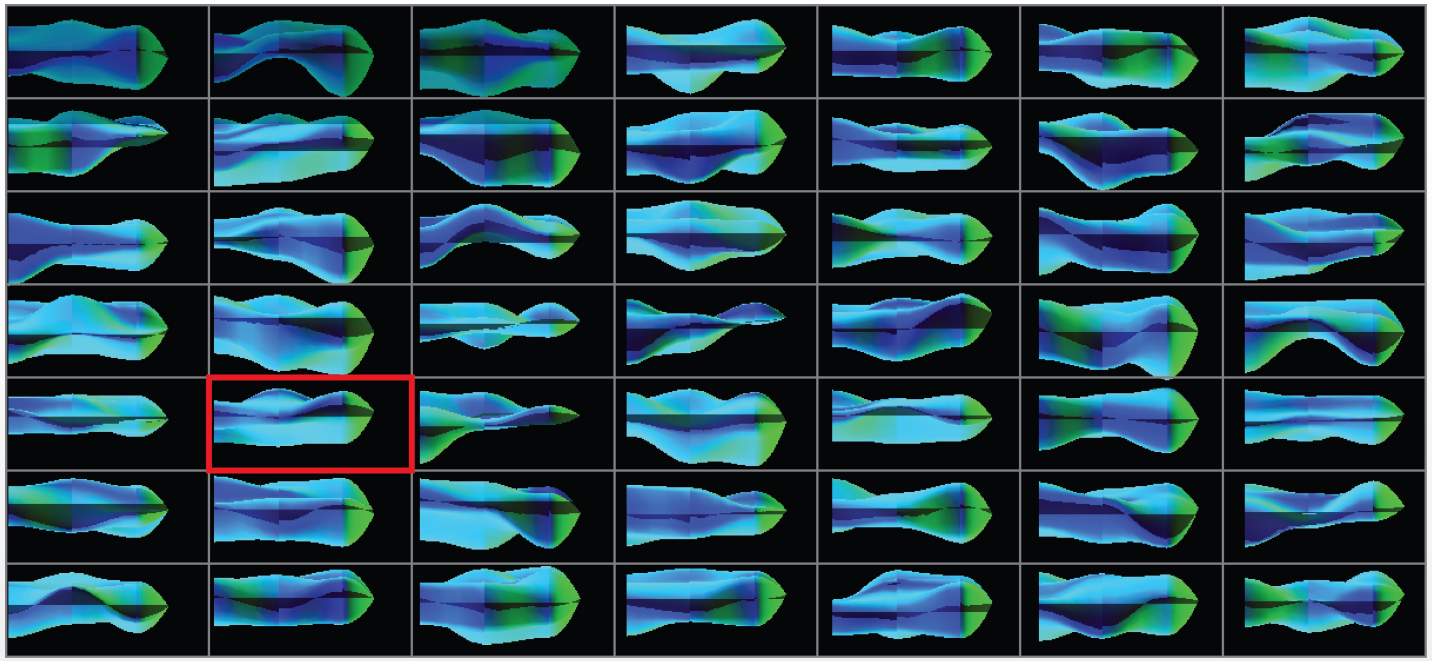




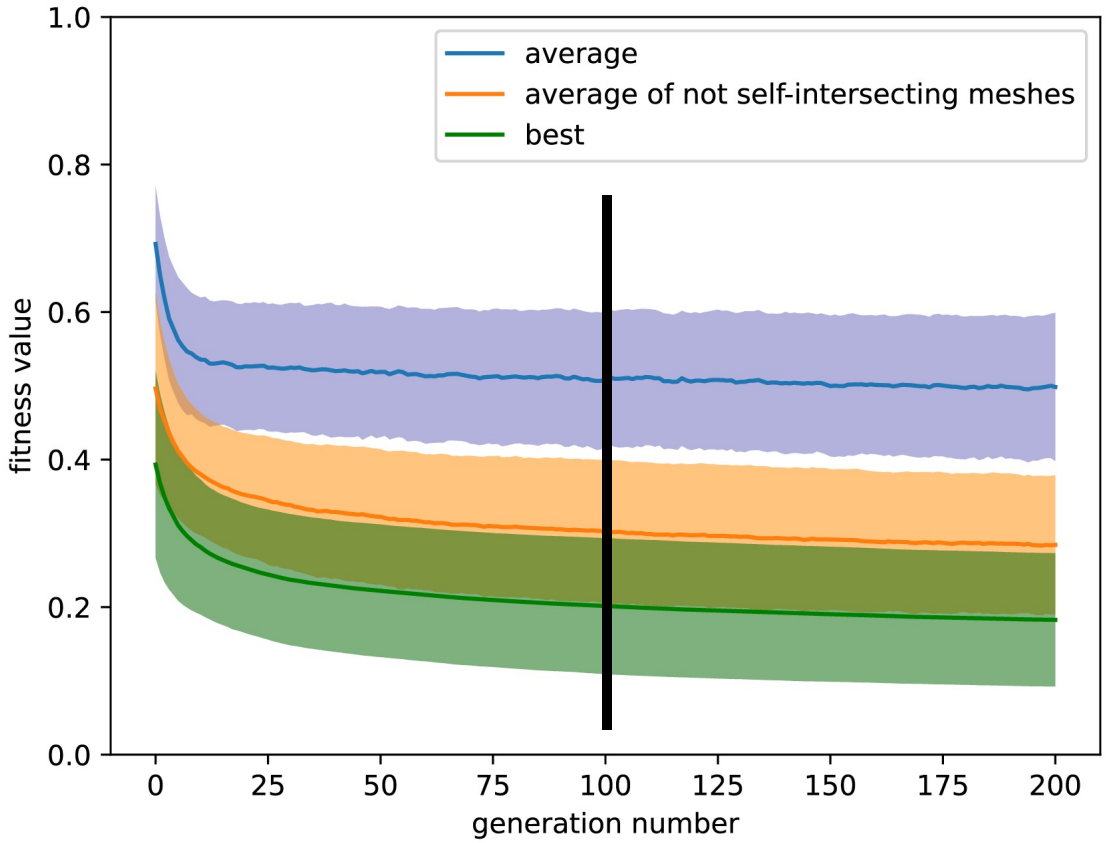


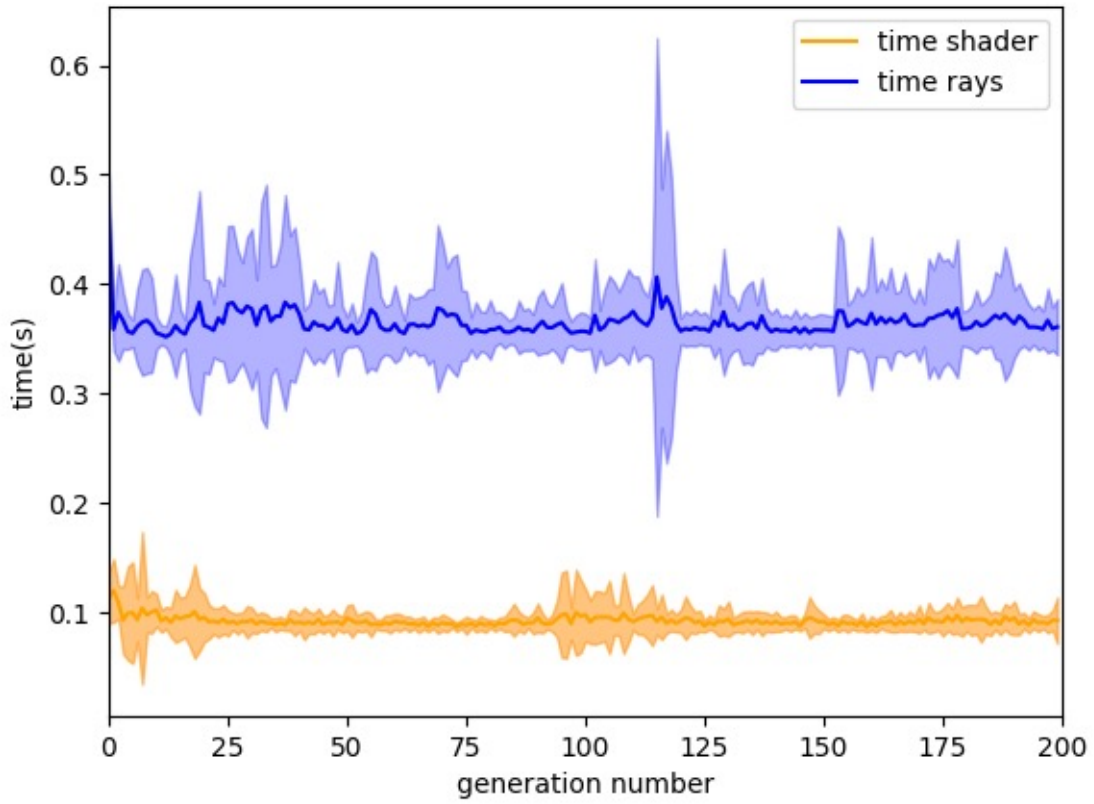
Proposed fitness function: approximated aerodynamics performance



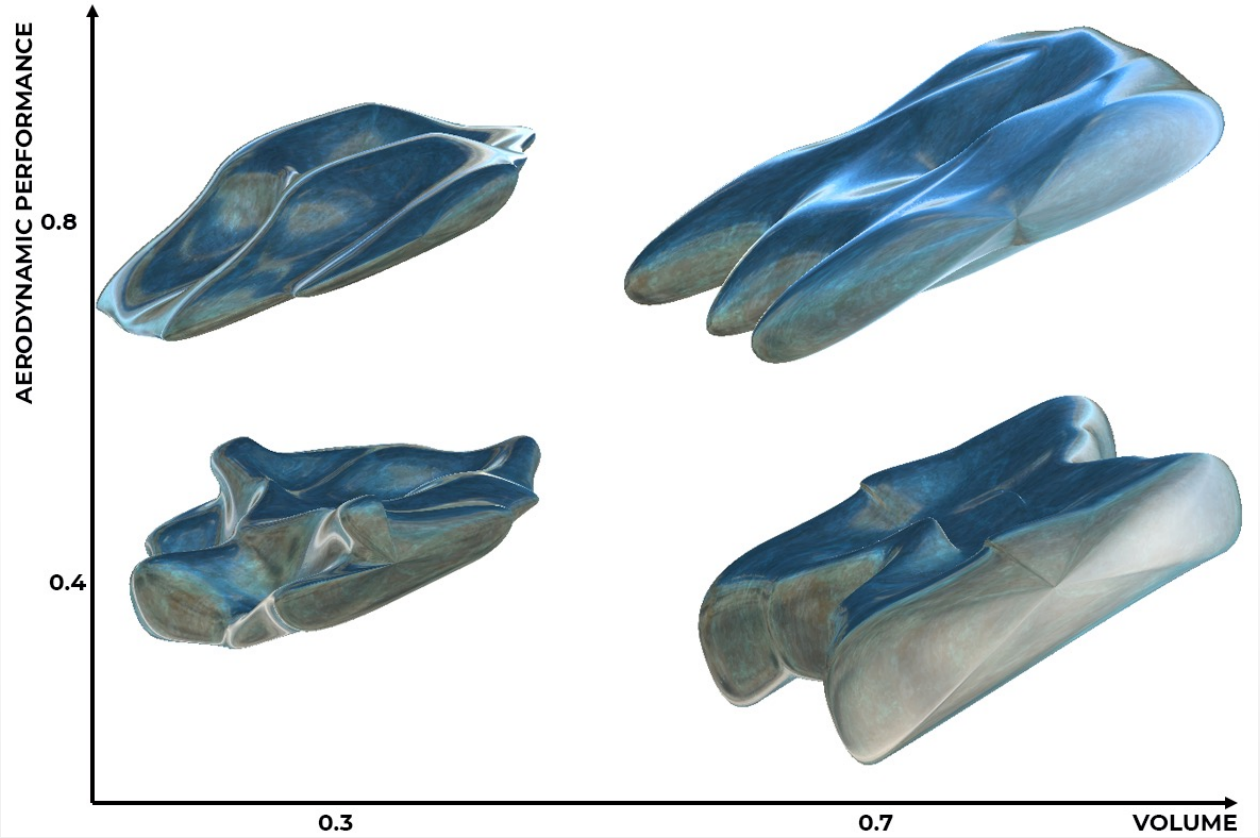


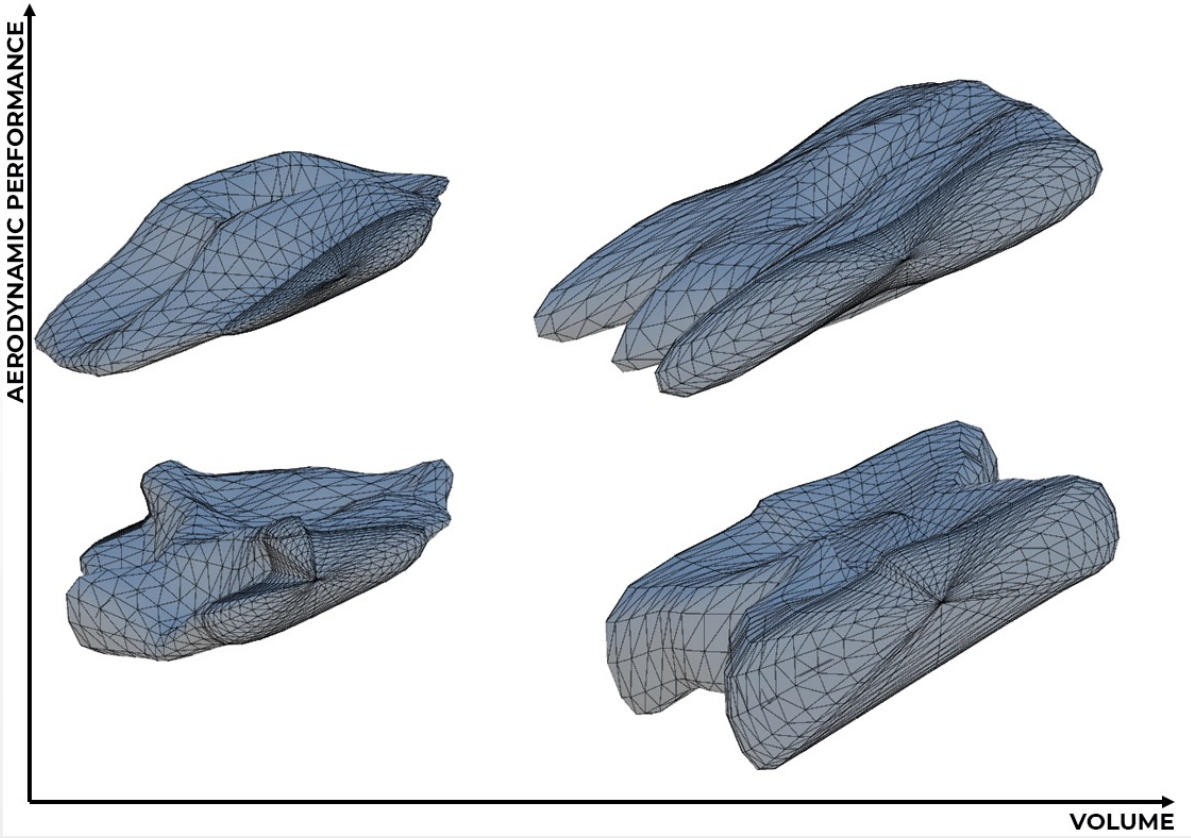
Proposed GA: performance



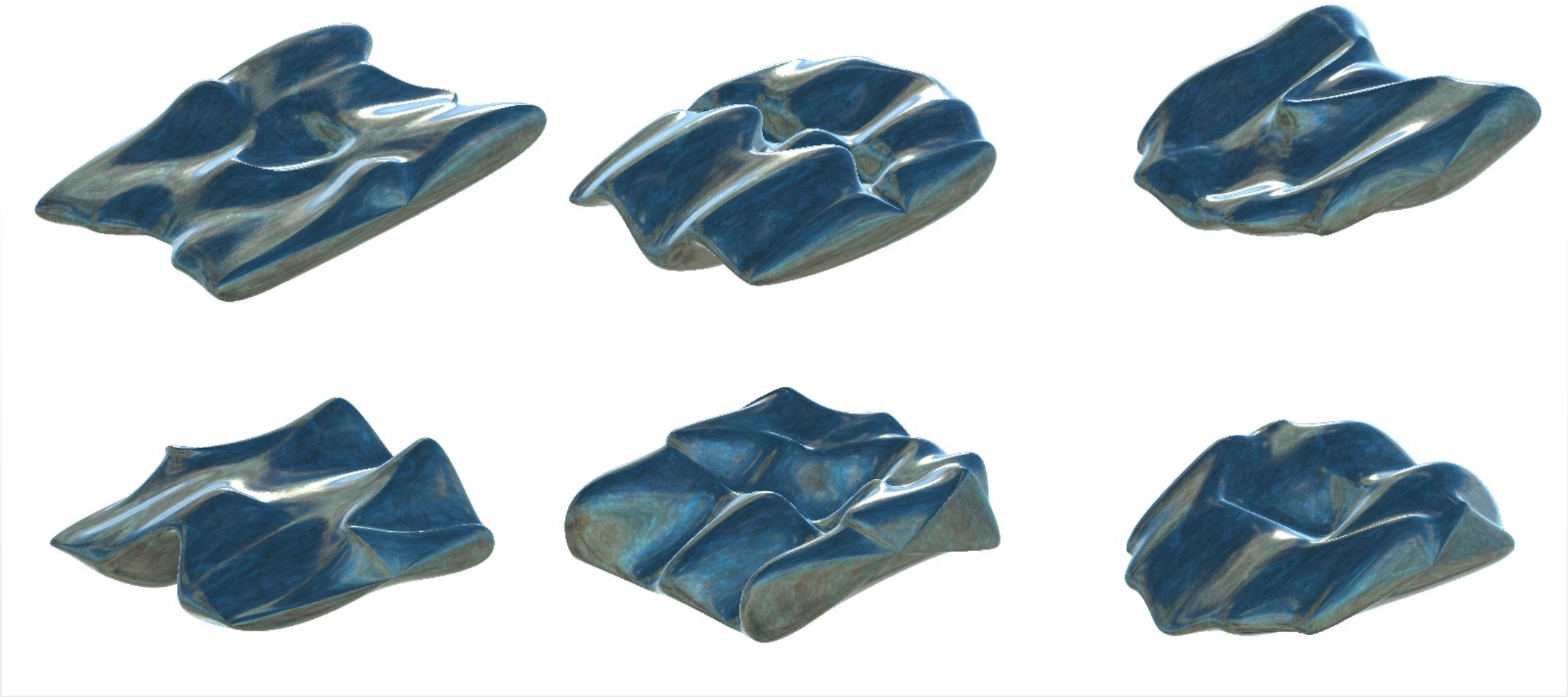


Proposed GA: results





Proposed GA: results



Proposed GA: results