Evolutionary-Based Procedural Content Generation for Videogames

Mike Mezhenin
Procedural Content Generation
Research Goal

**Goal** — implement evolutionary-based system to procedurally generate and adapt virtual characters in videogames

**Requirements:**
- Universal
- Adapts to user
- Generates diverse content
Evolutionary algorithm

Generate initial population ➔ Evaluate fitness of each individual ➔ Sufficient fitness achieved?

Result

Crossover ➔ Mutation

Crossover:

1111111111111111 ➔ 111111111100000000
0000000000000000 ➔ 111111111100000000

Mutation:

1111111110000000 ➔ 1110111100000000
Evolutionary algorithm

Generate initial population

Select individuals with most fitness

Evaluate fitness of each individual

User interaction with generated solutions

Crossover

Mutation
Representation

Character

1..*

Statistic

Name
Minimal value
Maximal value

0..*

Class

Name
Enumeration of values

0..*

Skill

Effect type
Affected object
Affected statistic
Effect power
Effect duration
<character>
  <attribute
    name="health"
    minValue="1"
    maxValue="100"
    step="1"/>
  <attribute
    name="strength"
    minValue="1"
    maxValue="10"
    step="0.5"/>
  <category
    name="race"
    values="elf,orc,human"/>
  <interactionTypes>
    onAttack,onDefend
  </interactionTypes>
  <objectTypes>
    self,player
  </objectTypes>
</character>

<character>
  <attribute
    name="health"
    value="10" />
  <attribute
    name="strength"
    value="5.5" />
  <category
    name="race"
    value="orc" />
  <skill
    type="onAttack"
    object="self"
    attribute="strength"
    value="5"
    duration="10" />
</character>
Implementation design

**Videogame**

- Game start
- Content generation
- Player interacts with the content
- Evaluating content
- [continue]
- [exit]

**PCG system**

- Generate initial population
- [new game]
- [saved game]
- Population
- Selection
- Crossover
- Mutation
- Saving population
- Ratings
- Content specifications
- Character traits
Evolutionary algorithm

Selection
  — tournament selection
  — absolute fitness replacement

Crossover
  — uniform

Mutation
  — normal distribution
Evaluation in a videogame
# Evaluation in a videogame

<table>
<thead>
<tr>
<th>Part</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Health</td>
<td>Body</td>
</tr>
<tr>
<td>Speed</td>
<td>Legs</td>
</tr>
<tr>
<td>Strength</td>
<td>Hands</td>
</tr>
<tr>
<td>Agility</td>
<td>Head</td>
</tr>
<tr>
<td>Perception</td>
<td>Eyes</td>
</tr>
<tr>
<td>Morale</td>
<td>Mouth</td>
</tr>
<tr>
<td>Luck</td>
<td>Hat</td>
</tr>
</tbody>
</table>

Health Evaluation in a videogame
Evaluation in a videogame

Crossover

+ 

Mutation

Fitness criteria

\[ f = \begin{cases} 
  -r + k \times d, & \text{player survived} \\
  -r - k \times d, & \text{player died} 
\end{cases} \]

\( r \) – distance between player and character  
\( d \) – damage inflicted by the character  
\( k \) – damage/distance balance coefficient
Evaluation in a videogame

Level 1

Level 4

Score

Health

Population #

Score

Health

Population #
Evaluation in a videogame

Health dynamics

Strength dynamics

Score  Health

Score  Strength

Population #
Character representation scheme suitable for different videogames was described.

PCG framework based on that representation was implemented.

Results were successfully evaluated.