

Generative Approach for Extending Computer Role-Playing Games at Run-Time

Juha-Matti Vanhatupa

Department of Software Systems
Tampere University of Technology

Contents

- Motivation
- Context: the CAGE game engine
- Basic Approach
- Prototype Implementation
- Generation Example
- Conclusions

Motivation

- Game world size of CRPGs has been a competitive advantage. Helped to sell games. -> the virtual worlds have grown in size over the years
 - 1) Application developers must use lot of time in programming game worlds
- Although game world is divided into areas(levels), large parts of game world are still loaded in one initialization operation
 - 2) loading times of CRPGs are rather long

Motivation

- Almost all state-of-the-art computer role-playing games use scripting languages
- Scripting languages are used for game world contents creating, artificial intelligence and user interface customization
- Scripting languages enable run-time generation

Context

- The CAGE Game Engine implemented during summer 2008.
- It is used as a platform in the game programming course of Department of Software Systems. Students implement their game projects using CAGE
- CAGE game world contents(non player characters and items) are loaded from Lua script files
- Excellent test platform for the approach

The CAGE Game Engine

- CAGE offers wide Lua interface
- Lua scripts can be used, for example, create creatures, items, set their behavior and properties
- C++ part of CAGE can also be extended
- Uses Crystal Space 3D engine for computer graphics

Lua scripts	C++ extensions to CAGE
CAGE core framework	
Crystal Space 3D engine	CAGE AI framework

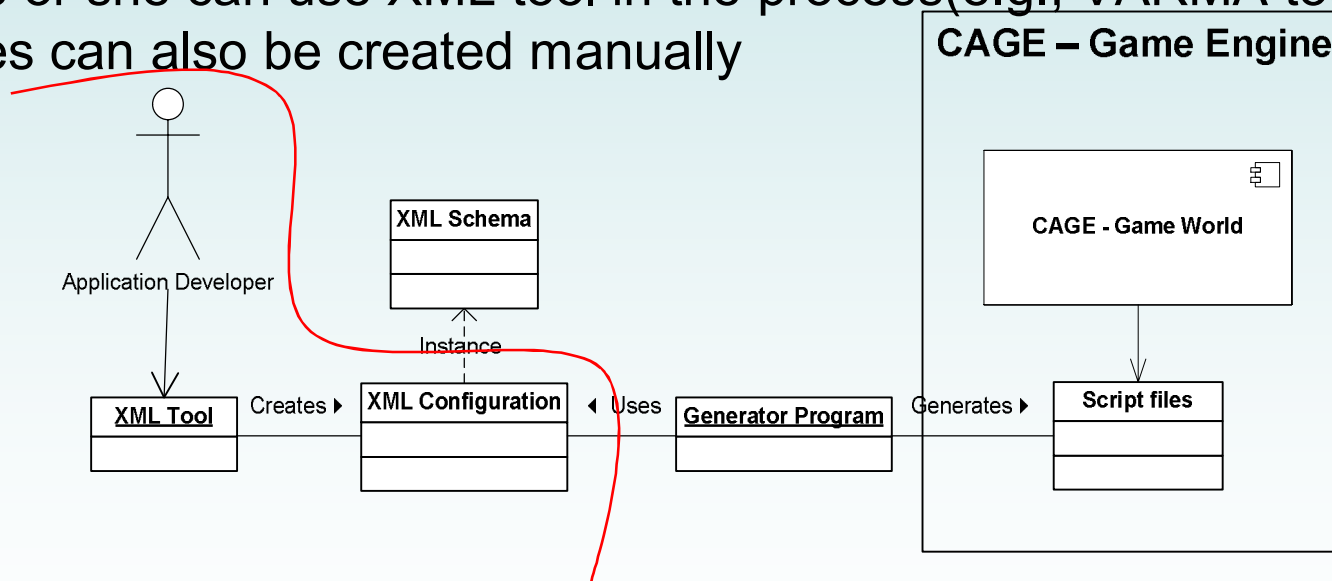
The CAGE Game Engine

- Supports state machines for AI programming (states implemented in Lua scripts)
- Does not limit created computer game type in any way



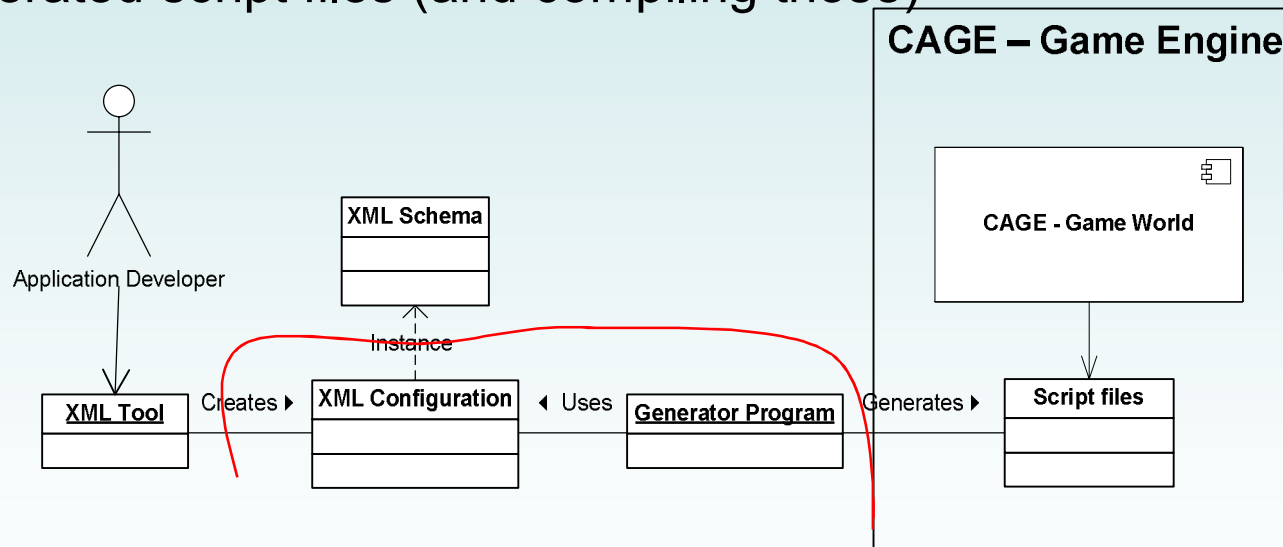
Basic Approach 1/3

- The application developer defines an XML file containing small piece of the game world
- The XML file contains limits for attribute values and numbers of objects, that will be created (object templates for generated objects)
- He or she can use XML tool in the process(e.g., VARMA tool), but files can also be created manually



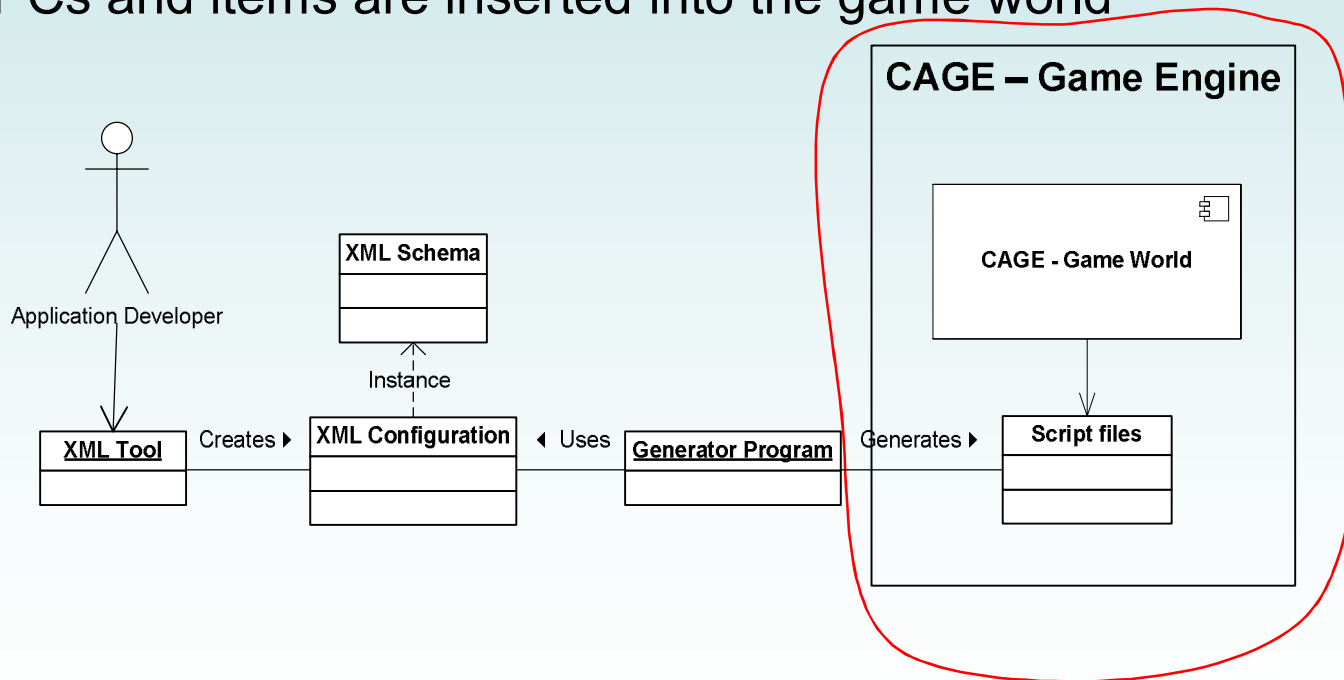
Basic Approach 2/3

- Object types are implemented in C++ or Lua
- Generation does not create new type of objects, it instantiates, configures and places objects
- Generator can also be run separately, which allows browsing of generated script files (and compiling those)



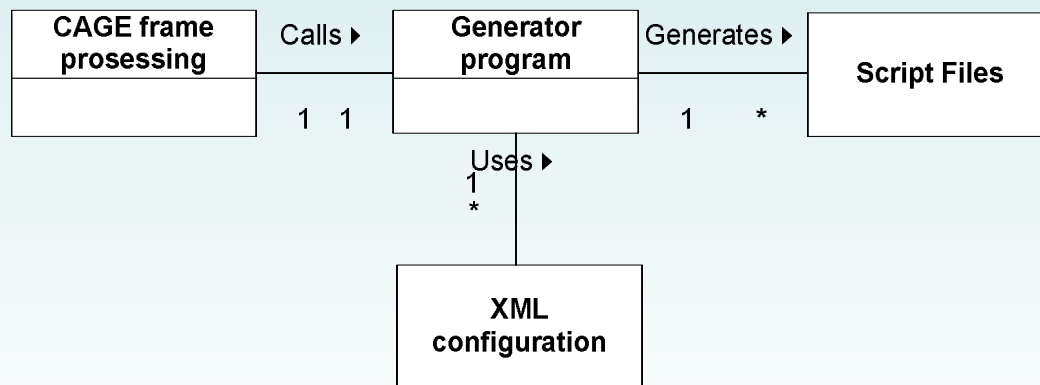
Basic Approach 3/3

- The player character advances into a new area of the game world:
 - the generator reads the XML file
 - generates script code based on the XML file
 - runs the script code
 - >NPCs and items are inserted into the game world



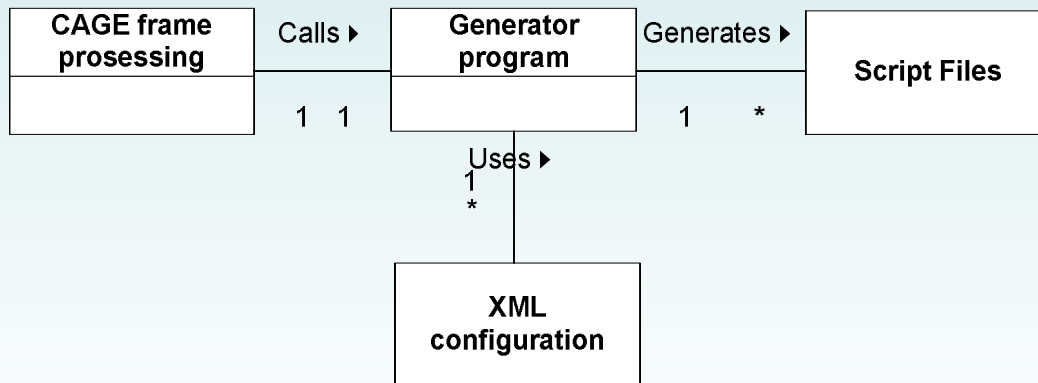
Prototype Implementation

- CAGE frame processing calls the generator, which checks if the player is in a new area.
- If the player has changed area, the generator reads the XML file, generates script for it, and runs the script.



Prototype Implementation

- Implemented using Lua and C++
- C++ code works only as a wrapper to call Lua functions



Example 1/3 The XML file

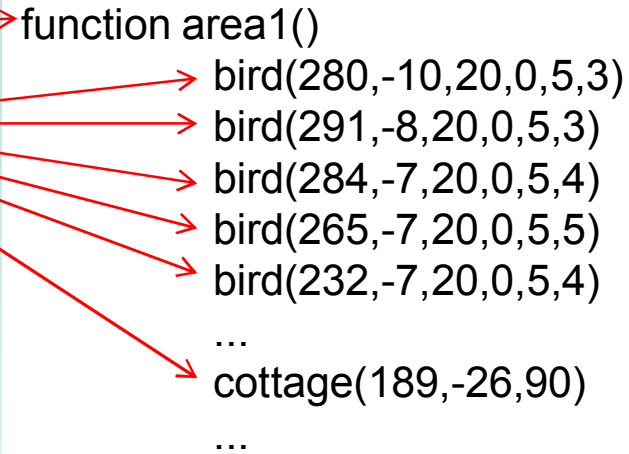
- Defines areas
- Object templates define objects

```
<area1>
  <!-- area size, if object does not define other x and y
  limits, it is generated inside area -->
  <areax>0</areax>
  <areaxe>300</areaxe>
  <areaz>-50</areaz>
  <areaze>300</areaze>
  <creature>
    <name>bird</name>
    <minimum>2</minimum>
    <maximum>10</maximum>
    <speed>5</speed>
    <hpmin>3</hpmin>
    <hpmax>5</hpmax>
    <xmin>230</xmin>
    <xmax>300</xmax>
    <ymin>-10</ymin>
    <ymax>-6</ymax>
    <z>20</z>
  </creature>
```

...

Example 2/3 The generated script

- Function / area
- Generated objects
- Attribute values
generated based on
attribute limits
- Generated code lines are
Lua function calls, which
call C++ constructor and
function to insert object
into the virtual world



```
function area1()  
    bird(280,-10,20,0,5,3)  
    bird(291,-8,20,0,5,3)  
    bird(284,-7,20,0,5,4)  
    bird(265,-7,20,0,5,5)  
    bird(232,-7,20,0,5,4)  
    ...  
    cottage(189,-26,90)  
    ...
```

Example 3/3 The result

- A flock of birds and a cottage is generated into the virtual world at the moment area border is crossed.



Generation times

- Case 1:
30 birds, 10 cowboys, 20 items
 - the whole operation 1.141s
 - script generation 0.016s
- Case 2:
1000 birds
 - the whole operation 1.922s
 - script generation 0.047s
- Test environment T60 laptop 2,0GHz with 1,00Gb RAM

Conclusions

- Approach can reduce loading times of computer role-playing games
- Can release application developers from boring, mechanical work
- Useful also in creating so-called wastelands

Future work

- Quest generation for computer role-playing games
- Scripting languages in game programming, comparison etc.
- Extending the CAGE artificial intelligence framework