Design and implementation of games based on existing graphics engines

Pawel Lekki[*]
Grzegorz Labuzek[†]

Institute of Computer Graphics
Szczecin University of Technology
Szczecin / Poland

Abstract

Creating computer games using already made engines has become very common practise. It is very difficult to create a technologically up to date and interesting game. It requires a large budget and a lot of time to catch up with the development of constantly new technologies. The solution to this problem is using a non commercial game engine license. Creating a game based on an existing engine saves us the trouble of implementing the basic functionality. We can then focus on expanding such an engine with additional modules and capabilities to create a new and technologically innovative implementation. This article describes the process of creating a game based on the Source Engine [20]. The D.I.P.R.I.P. Modification (`www.diprip.com`), developed by students of the Szczecin University of Technology, is shown as an example of this process.

Keywords: computer games development, 3d models, textures, real time graphics

1 Introduction

Computer game development in the present has become an increasingly complex process. The times when a single man could write a whole game are long gone because of the constantly increasing amount of technology and assets that need to be put into a game. Modern games include the most advanced systems for rendering, animation, sound, networking and physics simulation.

This brings out the problem of time and money that are needed to produce an entire game from scratch. Some companies buy ready made solutions like Karma or Havok physics engines [12], from third parties and other buy entire game engines like the Unreal, Quake, Source or Aurora [11, 13, 20, 3] engines. So is there still any place left for people who want to make games that are up to date with all of the current technology but do not have the money to hire a big professional team to create a modern game engine?

The solution is using a commercial game engine on a non commercial license. Games built using already published game engines are commonly called Mods (from the English word ”modification”) and are basically modifications of existing games. The amount of modification can range from only changing the clip size of a weapon in a first person perspective shooter, to creating completely new game assets, changing the game genre and adding various technology modules like a new particle or rendering system. Mods that result in a totally new game are commonly called total conversions.

This article discusses the process of creating a Mod using Valve Software's Source Engine [20] and gives an overview of the most important current game engines. As an example of a total conversion, the D.I.P.R.I.P. Mod is presented. It is a multiplayer game created by students of the Faculty of Computer Science at the Szczecin University of Technology. The group was leaded by the authors of this paper. The Mod is based on Valve Software's Source Engine [20].

The authors of this paper have taken active part in the Mod's creation. Pawel Lekki is responsible for the D.I.P.R.I.P. Game Design Document as well as two game environments. Grzegorz Labuzek is responsible for programming game rules and special effects.

The section 2 of this paper lists the most popular game genres and engines. It also describes the basic functions of the Source Engine [20]. The third chapter goes through all elements of the development process. It also explains possible differences between making games based on different engines. The fourth chapter shows the development process behind the D.I.P.R.I.P. Mod and the state of it's implementation. In the Conclusion section of this article a short summary of the current achievements and future works is given.

2 Background

Game MODs cover many game genres and existing titles. The most popular game genres are First Person Shooters (FPS) [20, 11, 13, 13, 5], Real Time Strategies (RTS) [18],

[*]plekki@wi.ps.pl
[†]glabuzek@wi.ps.pl

Turn Based Strategies (TBS) [1] and Role Playing Games (RPG) [3]. They use the best and most popular game engines that feature support for game Modding and working Software Development Kits.

However a game Mod is based on a specific game engine, thus a game genre, it can easily become a game of a completely different type. This Modding approach is described later on using the D.I.P.R.I.P. Mod example. The most commonly Modded game genre is the FPS and it is also the main scope of this article.

One of the most advanced and popular game engines [17, 8], currently available, is the Valve Software's Source Engine. Currently it has about 1100 registered Mods listed at the Mod Database website[1].

This engine supplies solutions for many key problems in creating a game. It is also very easily enhanced with additional modules. It gives the possibility to concentrate on creating new elements and not worrying about a basic implementation. The most important features of this engine are:

- Renderer:
  - DirectX shaders version 2.0 written in the High Level Shader Lanugage (HLSL) [10],
  - vertex lighting, light mapping, dynamic lighting [17],
  - High-Dynamic Range (HDR) lighting [19],
  - projected shadows.
- Materials system:
  - one material can be made up of multiple textures,
  - custom physics and sound properties can be assigned to materials,
- Multiplayer networking support:
  - Local Area Network (LAN) and Internet connectivity support,
  - Prediction analysis for interpolating collision/hit detection.
- Physics simulation:
  - constraint systems and ragdoll physics,
  - kinematic animated bone followers,
  - AI characters can interact with physically simulated objects.
- Sound:
  - Custom software sound effects - Digital Signal Processing (DSP),
  - MP3 decompression,
  - 5.1 surround sound.
- Artificial intelligence:
  - determines relationships such as friend/foe status of other entities,
  - intelligent character interaction during non-combat scenes.
- User interface:
  - Server browser,
  - VGUI (Valve's Graphical User Interface) - a universal set of tools for creating interactive user interfaces.

[1]http://www.moddb.com/

Some of the best non commercial Mods have gathered huge numbers of players and eventually they have been bought by the companies that produced the original games they were based on. The best example of such a mod is Counter-Strike, based on the Half-Life Engine. According to Valve Software's Steam distribution system statistics, it has an average of 100 000 players at all times and is the most popular multiplayer FPS.

3 Development of games based on existing game engines

This paragraph describes how a game Mod is created. It also goes through the key elements in this process and tries to give an explanation about their significance.

3.1 Development pipeline

The process of creating a Total Conversion using a modern game engine is very similar to creating a new game. It incorporates creation of 3D models, animation, textures [9], sounds and programming. All these assets are then gathered inside a 3D environment built in a dedicated map editor. The process of creating a game Mod is presented in Figure 8.

Schema (Figure 8) is a general representation of the MOD creation process. The first and most important part is the game project. Before the actual game development a Game Design Document should be created [15]. It describes key game aspects and gives a clear overview of the games goals and technologies that are going to be used. After creating the Game Design Document the process is divided into three major pipelines. All work concerning them can be done in parallel.

The path for creating materials divides into creating a base texture for the material and relevant shader code and material properties. Older game engines were usually limited only to creating a base texture without any additional information. Materials created in this part are used in texturing static game environment geometry and dynamic in-game 3D models.

The 3D modelling pipeline encompasses creation of highly detailed object meshes. This process also involves creating proper texture coordinates. Dynamic models also require creation of proper animations. After creating all components of a 3D model it is then prepared (compiled) to comply with an engine's standards.

The last major path is programming. This part includes creating game rules and game entities. Game mechanics are inseparable from the main code and usually take a lot of time for balancing. Programming game entities is simply creating instructions and behaviour for certain game characters, enemies or objects.

Game sound creation involves making custom sound tracks and sound effects. It is not considered a major part of the development process as it usually does not require engine specific tools.

The last and most important part of the process is creation of game environments. A game environment is a place where all the technologies meet together and are shown to the player. All previously created game assets have to be properly placed inside a game map where the action will take place.

3.2 Environment modeling

Environments are a key element for every game and this rule is also applicable to game Mods. Most of the modern game engines divide the process of creating maps to:

- world geometry modelling and texturing - creation of basic shapes inside a game editor on which other elements are then placed,
- placing highly detailed 3D decoration models created inside advanced 3D modelling programs,
- placing light sources - placing all lighting information,
- placing entities/actors - placing all game specific objects like enemies or scripted sequences controllers.

Complexity of map editors and the whole process differs depending on the game genre. For example building a map (area) inside the Aurora Engine Toolset [3] limits the creation of geometry to placing specific tiles from a tile set. Also building geometry inside RTS editors like the Command Conquer: Generals Worldbuilder [18] limits the creation of geometry to editing terrain elevation and texturing it.

The most technically advanced game genre is the FPS. Most map editors enable level designers to create basic geometry using primitive geometry like cubes, cylinders, etc. In addition to that game environments are decorated with highly detailed 3D models. Most FPS games have dedicated map editors but there are also a few universal map editors that cover a wide range of games, for example Quark [2] and GtkRadiant [16].

3D models like for example game characters or highly detailed environment decorations are created inside advanced 3D modelling programs. Specialised 3D modelling applications support features like rendering shadows onto previously created textures which can increase the visual quality of 3D models and create effects that could not be achieved in real time shadowing based from global illumination. They can also help by creating normal maps from highly detailed geometry and applying that onto lesser resolution models.

There are currently many sophisticated 3D modelling applications that use different file formats but most of the game engines have special exporters featured in their Software Development Kits for the most popular applications. The best modelling programs like 3DSMAX, Maya or XSI is usually not affordable for people who want to create game Mods. Fortunately there are good free alternatives like Blender or the XSI Mod Tool.

3.3 Changing game behaviour

The amount of programming changes is limited to the type of licence that is used. A full commercial engine licence enables access to the lowest level engine functions, thus full modification of the game engine is possible. Non commercial SDKs for the newest games never feature the lowest level engine code like for example rendering code or physics simulation code. A non commercial engine licence is usually limited to expanding the existing game engine with new modules and modifying high level parts of the game like game rules.

Although a non commercial license has it's limitations it is available at a very low cost, because it only requires the user to buy the game. It is usual that new engines base on old engines. It leads to the conclusion that by expanding an engine with new functions we can create something new and innovative. This eliminates the need to start from implementing the lowest level functions.

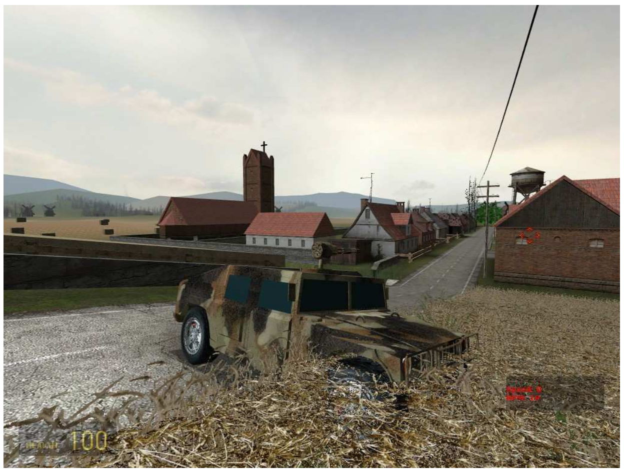

Figure 1: D.I.P.R.I.P. - an abandoned village

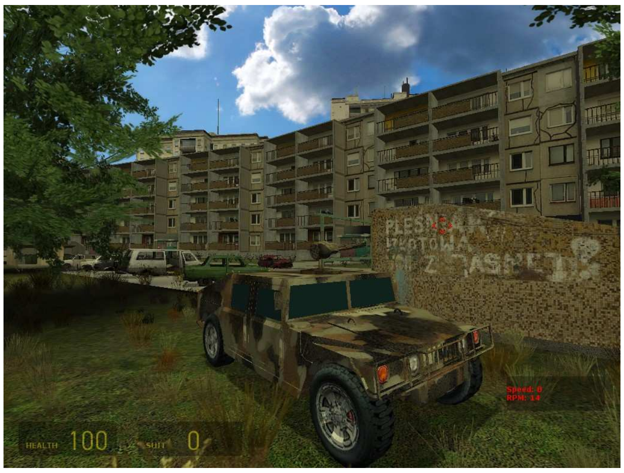

Figure 2: D.I.P.R.I.P. - an eastern European city district

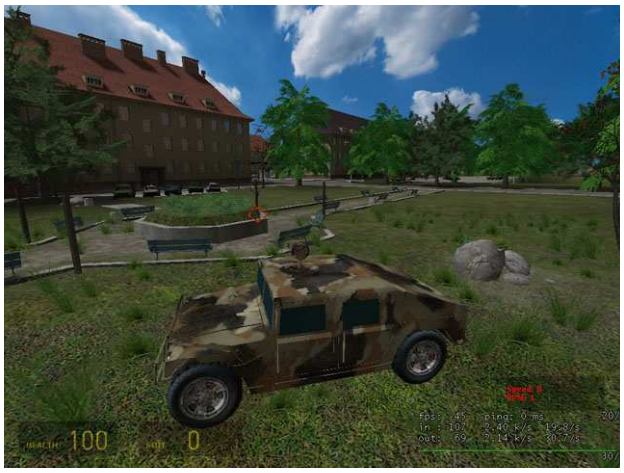

Figure 3: D.I.P.R.I.P. - the surroundings of the building of the Faculty of Computer Science at the Szczecin University of Technology

4 D.I.P.R.I.P. Implementation

D.I.P.R.I.P. is a multiplayer game Mod created by students of the Faculty of Computer Science at the Szczecin University of Technology. It is based on Valve Software's Source Engine [20].

The goal of the project was to create a commercial class multiplayer game utilizing the newest available technology with very limited funding and time resources.

D.I.P.R.I.P. is a multiplayer game were players fight each other using custom made armoured vehicles with various mounted weapons in modern post-war environments. Players observe their vehicles from a third person perspective and can not exit them.

The most important features in the D.I.P.R.I.P. implementation are: new environments based on real locations, new, upgraded vehicle implementation with networking support and design of a new user interface.

4.1 3D Environments and material system

3D game environments (maps) are a key element, on which the entire game action takes place. Three separate maps have been created: an abandoned village (Figure 1), an eastern European city district (Figure 2) and the surroundings of the building of the Faculty of Computer Science at the Szczecin University of Technology (Figure 3). All of these environments differ from each other what influences the process of their creation. All maps have been created using the Valve Hammer Editor.

Real life photographs were essential to the process of creating realistic looking environment objects. They were used to evaluate proportions of elements and to acquire realistic textures (see Figure 4).

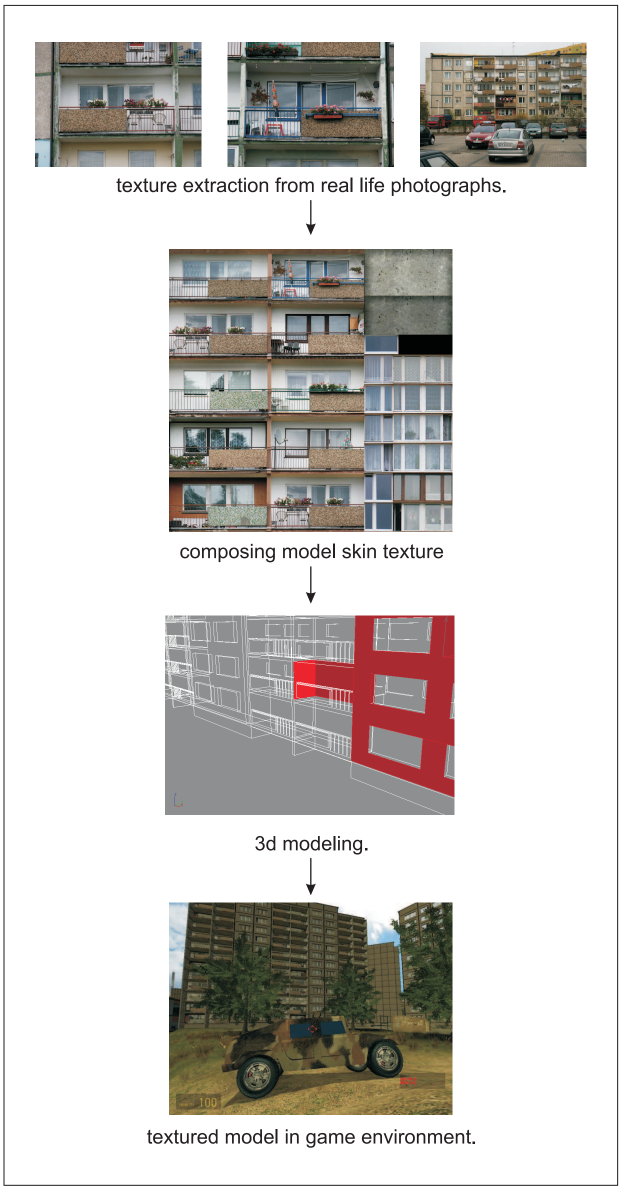
texture extraction from real life photographs.
composing model skin texture
3d modeling.
textured model in game environment.

Figure 4: Model creation process

4.2 Vehicles and weapon system

The vehicle weapons system from the original Half-Life 2 game has been completely changed to fit the requirements of the new game. The original system limited vehicles to have only one weapon and there was no modular solution similar to that used with human characters that would enable players to switch between weapons.

That required an implementation of a completely new modular solution that would fit the vehicles. The new system enables vehicles to have weapons attached to separate mounting points with separate animation control and special effects. It also created a need to change the vehicle modeling and animation system. The number of skeletal animation files required to fully animate a vehicle has increased proportionally to the number of new weapons (currently four).

Pictures 5, 6, 7 illustrate the process of creating a vehicle concept drawing followed by a highly detailed model and an ingame model.

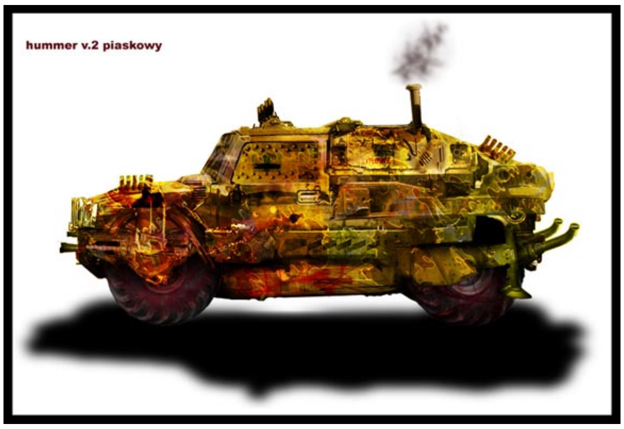

Figure 5: Model creation process: The Hummer - a vehicle concept art drawing

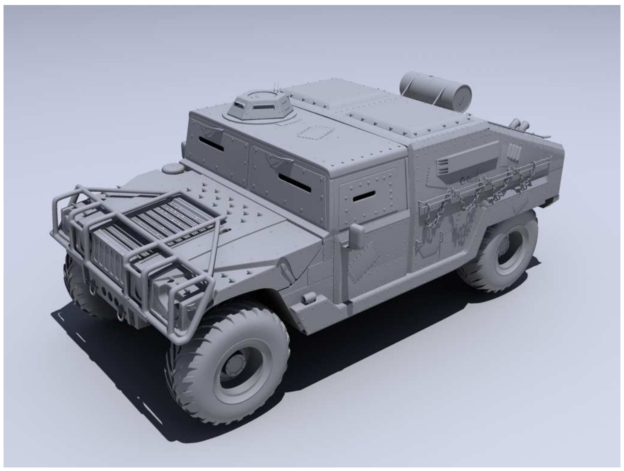

Figure 6: Model creation process: The Hummer - a highly detailed model

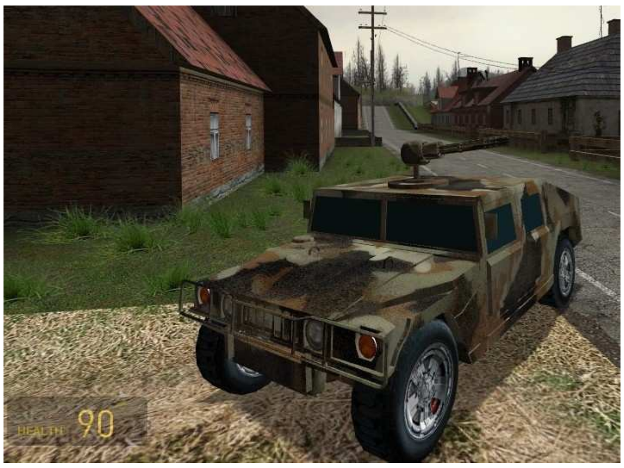

Figure 7: Model creation process: The Hummer - an ingame model

4.3 Networking

A lot of upgrades in the networking system concerning vehicles have been implemented in D.I.P.R.I.P. The original network code did not have any optimizations for vehicles. It was not possible to play a multiplayer game based on vehicle combat without that. It was necessary to shift a lot of code from the server to the client. This operation saved most of the bandwidth used by a vehicle. The D.I.P.R.I.P. team created a new class for a player which combined some functionality of the original player class and the old vehicle class and that also saved some communication between the server and the client. There have also been changes in prediction tables which had to be updated by new variables added in the new player class. More about Source Engine networking can be read here [4].

4.4 User Interface

The user interface in D.I.P.R.I.P. is based on VGUI (Valve Graphics User Interface). VGUI is an implemented set of commonly used elements like: windows, buttons, editable fields, menus etc. It is a very flexible system which allows the creation of a complex interface with minimum effort. Programming a user interface is mostly event based. VGUI is able to use True Type fonts and render them in a desired location as a texture. For the needs of the D.I.P.R.I.P. Mod a new heads up display has been created using the VGUI. It contains vehicle statistics which are used by the player. The new user interface also has a special panel which is used to change vehicles during the game.

5 Conclusion and future work

The D.I.P.R.I.P. Mod for the Source Engine shows that it is still possible to create a modern game using very limited resources. It is a completely different game from the one that it is basing on - Half-Life 2. Among the implemented features are a new third person camera, the change from a human player character to a player controlled vehicle and a new vehicle weapon system. Game maps have been created in a different scale to fit the need for big open spaces and semi realistic environments. The Mod also features custom made models and textures which have been created using real life photographs. New player controlled vehicle models with physics behavior scripts.

Future works will include new shaders written in the HLSL shader language, improved vehicle physics, an algorithm for recovering stuck vehicles and additional network code optimizations. New user interface elements like a map/radar showing the player and enemy positions will have to be created. Additional game assets including new vehicle models, sounds, textures and maps.

Game engines can also be used to make ”serious” MODs. The Source Engine is also being used to create a medical simulation where players can learn human anatomy or perform virtual operations [14, 7]. Another example of it's use is a forestry statistics simulation [7]. Game engines can also be used to create scientific simulations by implementing AI learning algorithms to for example game enemies or wild animals in an artificial environment [6].

Another good implementation of using a game engine can be a virtual museum where the player could see places which are normally inaccessible because of hazardous environment or the possible danger that a tourist can create to a monument. One of such projects has been started at the Institute of Computer Graphics at the Technical University of Szczecin. It focuses on virtuall recreation of the Wieliczka Salt Mine in Poland which is a monument of the UNESCO World Cultural Heritage List.

These examples show that Mods can be created not only for the purposes of entertainment. They can be also used for educational projects and as help for scientific simulations. Using an existing graphics engine can substantially decrease the time needed to finish a project. The idea of using existing graphics engines can help in the development of many projects that would otherwise be very difficult to accomplish.

6 Acknowledgements

We would like to thank all members of the D.I.P.R.I.P. team that did not participate in the writing of this article:

- Wojciech Lekki - project management, 3D modeling, textures, environment modeling
- Mateusz Kotiuk - programming
- Wojciech Wozny - programming
- Krzysztof Miler - 3D modeling
- Konrad Wyremski - concept artwork
- Tomasz Tulikowski - environment modeling
- Pawel Stelmach - sound

References

[1] Multile authors. Civ4 - creation customization. http://forums.civfanatics.com/forumdisplay.php?f=158, visited: 4.02.2006.

[2] Multiple authors. Quark. http://quark.planetquake.gamespy.com/, visited: 4.02.2006.

[3] BioWare. Introduction to the aurora neverwinter toolset. http://nwn.bioware.com/builders/toolsetintro.html, visited: 4.02.2006.

[4] The Valve Developer Community. Source multiplayer networking. http://developer.valvesoftware.com/wiki/Source_Multiplayer_Networking, visited: 4.02.2006 22:00.

[5] Crytek. Cryengine specifications. http://www.crytek.de/technology/, visited: 4.02.2006.

[6] Mark DeLoura. *Game Programming Gems*. Charles River Media, Inc., Rockland MA, 2000.

[7] Rusel DeMaria. Sgs 2005: Healthcare and forestry half-life 2: Meet serious games modding, 2005. http://www.gamasutra.com/features/20051103/demaria_01.shtml, visited: 4.02.2006.

[8] David Eberly. *3D Game Engine Design: A Practical Approach to Real-Time Computer Graphics*. Morgan Kaufmann Publishers, San Francisco CA, 2000.

[9] David S. Ebert, F. Kenton Musgrave, Darwyn Peachey, Ken Perlin, and Steven Worley. *Texturing and Modeling: A Procedural Approach*. Morgan Kaufmann Publishers, 2002.

[10] Randima Fernando. *GPU Gems*. Addison-Wesley, 2004.

[11] Epic Games. Unreal engine. http://udn.epicgames.com/Main/WebHome, visited: 4.02.2006.

[12] Havok. Havok dynamics. http://www.havok.com/, visited: 4.02.2006.

[13] id Software. id software's technology licensing program. http://www.idsoftware.com/business/technology/, visited: 4.02.2006.

[14] Jorge A. Ramirez Klaudia Johnston. Texas a m university-corpus christi awarded $4.3 million grant to develop virtual learning space for current and future healthcare professionals, 2005. `http://kanga.tamucc.edu/PublicAffairs/press/2005/june/nursing/`, visited: 4.02.2006.

[15] Tom Meigs. *Ultimate Game Design: Building Game Worlds*. Brandon A. Nordin, 2003.

[16] Tristan J́heraxB́lease / multiple authors. Gtkradiant. `http://www.qeradiant.com/`, visited: 4.02.2006.

[17] Tomas Mller and Eric Haines. *Real-Time Rendering*. A. K. Peters, Ltd., Natick MA, 1999.

[18] EA Pacific. Command conquer generals: Worldbuilder, 2003. `http://www.generalsmaps.net/worldbuilder/worldbuilder.pdf`, visited: 4.02.2006.

[19] Erik Reinhard, Greg Ward, Sumanta Pattanaik, and Paul Debevec. *High Dynamic Range Imaging*. Morgan Kaufman, Elsevier, 2005.

[20] Valve Software. Source engine licencing, information sheet. `http://www.valvesoftware.com/SOURCE_InfoSheet.pdf`, visited: 4.02.2006.

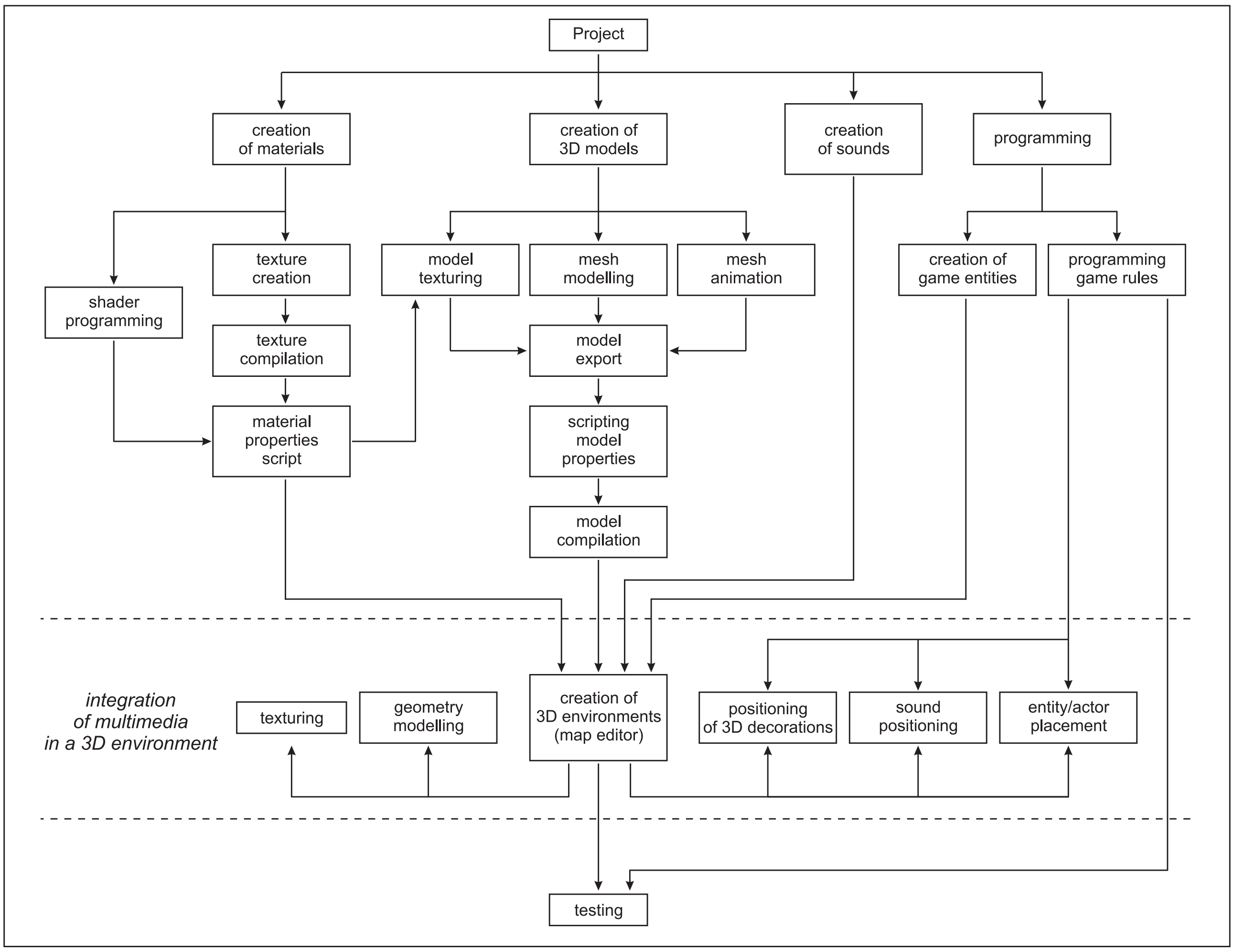

Figure 8: MOD creation process