# Content Creation for a 3D Game with Maya and Unity 3D

Matthias Labschütz[*] Katharina Krösl[†]
In alphabetical order: Mariebeth Aquino[‡] Florian Grashäftl[§] Stephanie Kohl[¶]
*Supervised by: Reinhold Preiner*

Institute of Computer Graphics and Algorithms
Vienna University of Technology
Vienna / Austria

## Abstract

'Dynamite Pete'[1] is a 3D game we developed with Autodesk Maya and Unity 3D in a team of 26 computer science students with varying skills and expertise in content creation. A game development pipeline explaining the production of the game from concept to release is presented. In addition, this paper explores the challenges faced by the project staff and outlines the experiences gained during the project's implementation.

**Keywords:** content creation, content creation pipeline, game production pipeline, Autodesk Maya, Unity 3D

## 1 Introduction

Nowadays, the quality of graphics and realism of games is constantly increasing, since consumers are always demanding a more realistic look and feel in their games. This means that improvement of renderings, outstanding content, more believable animations and more authentic behavior of artificial intelligence are needed. Therefore the work of artists and animators is crucial for the success of a game and the prosperity of a game development studio.

In 2010 a group of 26 students attending 'Maya-Course 2' at the Vienna University of Technology performed a game production process in a game development studio. The ultimate goal of this exercise was to produce a video game. Unity 3.1 [10] was chosen as game engine, but the focus was placed on 3D content creation for real time application, using, besides other tools, Autodesk Maya 2011. The game development team brought in different sets of skills which ranged from beginners with hardly any knowledge about content creation to students with particular profession. Every member of the team was given at least one role and had to fulfill tasks corresponding to their job title. The assigned positions covered Project Management, Technical Direction, Art Direction and different Artists. The main areas of these artists were Modeling and Sculpting, UV-Layout, Texturing, Lighting, Rendering, Rigging, Animation, Level-design and Sound. The resulting game 'Dynamite Pete' is a comic-style Western-Adventure where the player plays the role of the antihero named Pete and has to escape from a canyon. Figure 1 shows an example screenshot of the finished game.

Figure 1: In-game screenshot of the game 'Dynamite Pete'.

This paper explains a basic strategy for the professional development of a content-intensive video game in a large team and shares experiences from our project. In the following, we will shortly introduce the tools that were used (Section 2), schematize the workflow of our game development process (Section 3) and finally explain some selected challenges that had to be overcome during the creation of 'Dynamite Pete' (Section 4).

## 2 Tools

The main tools used throughout the project were Maya for modeling, animating and rendering and the Unity game

[*]Technical Director Workflow: e8971103@student.tuwien.ac.at
[†]Artist: e0325089@student.tuwien.ac.at
[‡]Producer: e0326746@student.tuwien.ac.at
[§]Art Director: e0300310@student.tuwien.ac.at
[¶]Technical Director Unity: e0626088@student.tuwien.ac.at
[1]http://www.cg.tuwien.ac.at/maya/

engine for implementation. In addition, image, audio and video editing tools as well as drawing, sculpting and communication tools were used.

### 2.1 3D Animation and Modeling Software (Autodesk Maya)

Autodesk Maya (freely available for educational purposes[2]) was chosen as 3D modeling, animation and rendering tool in this project. Maya provides artists with an end-to-end creative workflow [4]. As a professional tool it is very complex and offers a great number of features. Despite a steep learning curve, using Maya in its whole complexity requires a long-winded learning process. Another drawback is the lack of forward compatibility, which means all the artists had to work with the same version of Maya irrespective of their preferences.

### 2.2 Game Engine (Unity 3D)

In this project, the free version of the game engine Unity 3.1 was chosen for the production of 'Dynamite Pete'. There are different export options in this game engine, each one dedicated to another platform (e.g.: Web Player, PC and Mac Standalone, iOs, Android, Xbox 360, PS3), which eases the development for different consoles or devices. Unity attracts especially small and middle-sized development studios who hardly invest in expensive high-end rendering engines. Furthermore prototyping and game development is very quick due to the WYSIWYG ("what you see is what you get") editor which allows instant changes and live editing. In addition, Unity provides multiple built-in shaders and effects as well as a physics engine and collision detection.

For a students' project the most important reason for choosing Unity is the fact that it is very easy to use and to learn. Developing with Unity is mainly based on drag and drop with occasional adapting of scripts rather than writing code. Apart from shaders and effects which can simply be turned on in some game settings, Unity provides numerous scripts which can be dragged onto 3D models. These scripts act for example as character controllers, follow up cameras or other important features. However, Unity lacks of integrated modeling abilities, which is the reason for using Maya as external modeling tool.

Another major drawback of the free version of Unity is its lack of SVN support. This makes it difficult for multiple programmers to work concurrently on one single project, which will be discussed in detail in section 4. Like Autodesk Maya, Unity is not forward compatible, but overall the advantages outweigh the disadvantages. However, the use of Unity Pro is advisable, due the previously mentioned restrictions in the free version.

[2] http://usa.autodesk.com/maya/trial/

## 3 General Game Production Pipeline

A game production pipeline is basically a concept of workflow management for use in the game development process. The phases of this pipeline are certain tasks that need to be fulfilled until the release of a video game.

Figure 2 gives a detailed overview of the development process of this project, as it will be explained in the following chapters, showing the main five tasks in color.

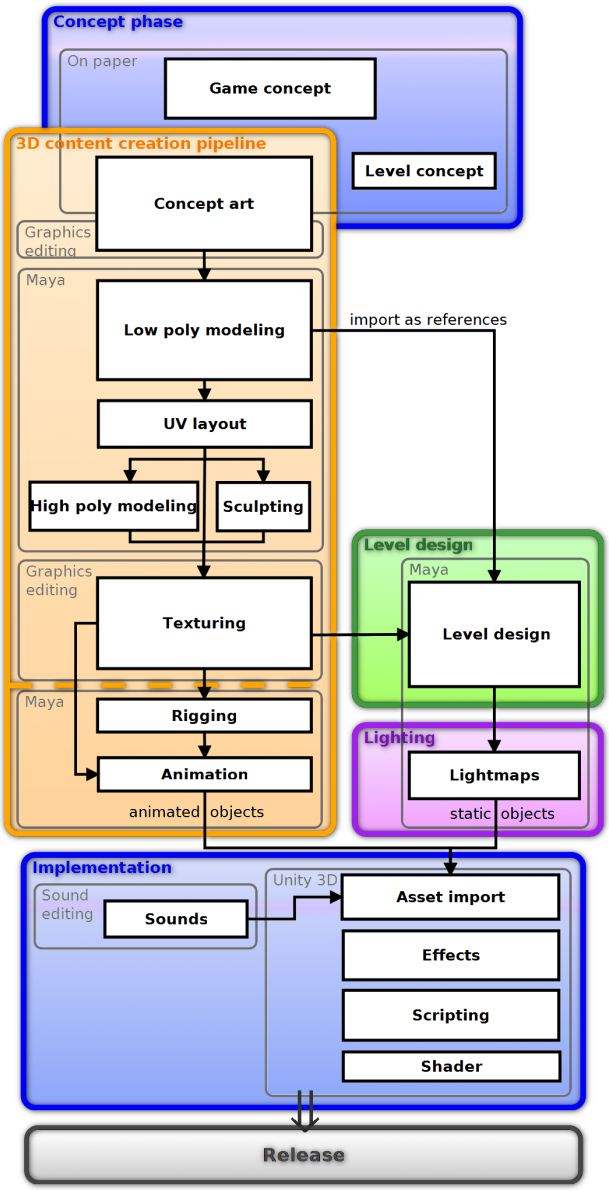


Figure 2: Game production pipeline overview: Concept, content creation pipeline, level design, lighting and implementation as organized in this particular project.

Some of these tasks require to be carried out sequentially as presented in the following. Nevertheless, to increase productivity, it should be a goal to break up this sequential approach, to allow people to work in parallel on different tasks.

Since this very project was an attempt to implement the workflow in a game development studio at a larger scale with very limited production time, it was necessary to follow a pipeline approach without iterations. The scope of this project was 3D modeling, with focus on content creation. Other aspects, like game-design, were of minor interest. The following paragraphs describe the production stages of the project in detail.

## 3.1 Concept Phase

During the concept phase a small team drew an outline of the plot, the setting and the game mechanics. The game's environment was chosen to be a wild-west desert. A comic style was preferred over a photorealistic style and the goal of the game was set to collecting dynamite for breaking out of a hostile canyon.

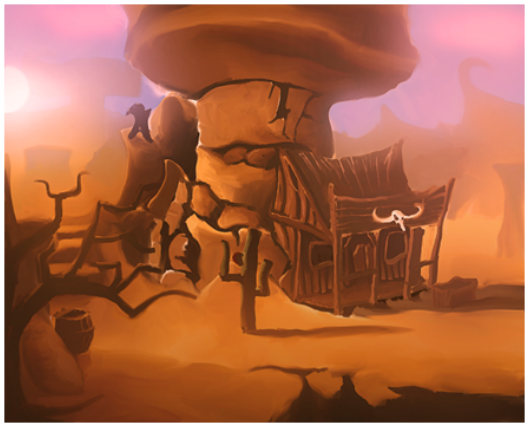

Figure 3: Early painted concept art

After this stage, the look and feel of the game content had to be worked out, requiring multiple revisions of concept arts. An example is shown in Figure 3. Inspiration was coming from all different sorts of media, starting from similar computer games and movies to comics and music. Color and proportions play an important part in the game's visual style. A dirty textured comic look was chosen, inspired by 'Star Wars - the Clone Wars Series' [9] and Woody, the Cowboy in 'Disney Pixar's Toy Story' [7].

## 3.2 3D Content Creation Pipeline

The orange box in Figure 2 illustrates the 3D concept creation pipeline which contains the stages every 3D model has to pass from concept art to the final model. The following paragraphs describe these stages in detail.

### 3.2.1 Concept Art

Figure 4: Character concept art of a sheriff and a barmaid model.

Concept artists worked primarily on the main assets of the game (characters and environment), gradually enhancing their work guided by feedback of the art direction and other artists. After the initial sketches on paper, some artists moved on to graphics editing tools, using tablets as input devices. As final step of the concept phase, they created colored front and side views of the assets, which modeling and texturing artists used as guides for their geometry and coloring. Figure 4 shows a small selection of colored character concepts. Most of the concepts were hand drawn and can be found on the development blog.[3]

### 3.2.2 Low Polygon Modeling

The first step from concept to 3D model is to create a low polygon model. Since Unity can handle triangles and quads, there was no restriction to use a certain modeling technique only. Nevertheless, most modeling artists preferred quads over triangles. Low polygon modeling also includes smoothing or hardening normals.

To avoid models with very differing level of detail, it is highly recommended to define a maximum polygon count in advance, depending on a model's size and importance. However, multiple revisions were necessary during this phase to achieve a consistent style for all the low polygon models.

For houses, a modular system was used to create numerous different house types out of individual basic parts. This tile based approach allowed more variations with less time for modeling.

### 3.2.3 UV Layout

Before texturing or high polygon modeling could start, UV layouts had to be done either by UV layout artists or by the modeling artists themselves. A UV set in Maya consists of

[3] http://twoday.tuwien.ac.at/mayakurs22010/

a single UV layout. An object with multiple texture types can have multiple UV layouts. For this project a maximum of three UV sets per object was defined:

- Color map UV set: For color textures. Faces can reuse texture space, meaning that a certain space in texture space can be used by multiple faces.
- Normal map UV set (optional): Only used if a separate normal map was required.
- Light map UV set: Can not include overlapping faces in texture space. Since each face can have different lighting information, each face has to have its own amount of space in the UV layout.

#### 3.2.4 High Polygon Modeling

After the UV layouts were done, some of the low polygon models were improved to high-resolution models either by a smooth-operation provided by Maya (adding additional subdivision polygons), or by artistic sculpting using Autodesk Mudbox [5]. The result of this high polygon modeling step was a normal map.

#### 3.2.5 Texturing

Texturing started right after the UV layout was done. A color table (Figure 5) was used as reference for texture artists. A screenshot of the UV layout in texture space was exported to digital image processing tools. The texture coordinates (UVs) served as orientation for texture artists, who painted their textures on these coordinates, using only colors from the given color table.

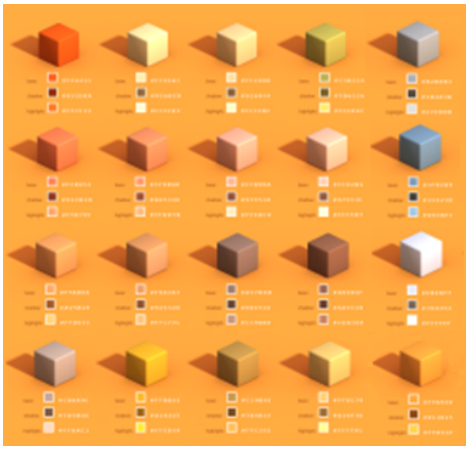

Figure 5: Color table for texture artists

For frequently used textures (such as wood), texture templates were created. For each contained material, these libraries provided bump maps, normal maps and differently colored layers which artist could switch on or off, to create new textures fast and easy.

#### 3.2.6 Rigging and Animation

To enable artists to easily create naturally looking animations of characters, a technique called 'rigging' is used, which creates a virtual bone structure for each character. This structure can then be used to control the movements of the characters for animation in a natural way. After the bone structure is finished, the character's mesh is bound to the skeleton and weighted between influencing bones. Usually, inverse kinematics are used to specify the bones position when moving handles of the skeleton.

In this project, each game character went through an individual rigging process in Maya, offering different controls for the animators. Since this was an educational project, most of the participants decided not to use premanufactured rigs (such as the built-in Full Body IK [3]). The animation artists then produced key-frame based animations for some movements (e.g.: walkcycle, idle, attack) for each game character, which were done in Maya.

### 3.3 Level Design

#### 3.3.1 The Level Design Process

The previous section described how single pieces of static and dynamic content have been created. The following sections will explain how all these parts are combined to form a virtual environment and finally a playable game. The first step on this path is level design which can be seen as an assembly process that incorporates or creates content to shape the environment of a game. Depending on the genre of a game, the scope of level design may lie on the content creation, the technical assembly, or the game-play tuning aspect.

**Scheduling:** In general, the crucial amount of work in level design is carried out at a late stage in the creation process. Not only because level design requires parts of the engine to be finished but also assets to work with. In order to allow the level design process start early, it is good practice to prioritize the assets depending on their importance for the level designer.

**Concurrent working:** Concurrent working on game levels is difficult to be done efficiently and can easily result in a high organization and merging overhead [11]. For small game environments, one person working on the level design is advisable to reduce the overhead. For larger projects, concurrent work of several designers on levels of high complexity can be achieved by splitting the levels along certain regions.

**Asset placement:** Open terrain game levels often require a lot of environmental objects to be placed into them to produce a rich and exciting game feeling. Automatic placement can help speeding up the process, while keeping a human designer in charge of where objects are placed. For

large projects, more advanced level-editing tools can be used, which allow modifying the layout in a more abstract and faster way (e.g. drawing a map of the area, while the content is placed automatically).

Figure 6 shows the level design approach in relation to other tasks during the creation of a game. The solid lines describe the path of content as it is passed through the team during the game development process. The dashed lines denote requests and tools which will not be included directly in the final release. As shown in Figure 6, level design is a step between content creation and programming. Therefore, changes in level design require interactivity or fast compiling mechanisms to see the impacts on the game.

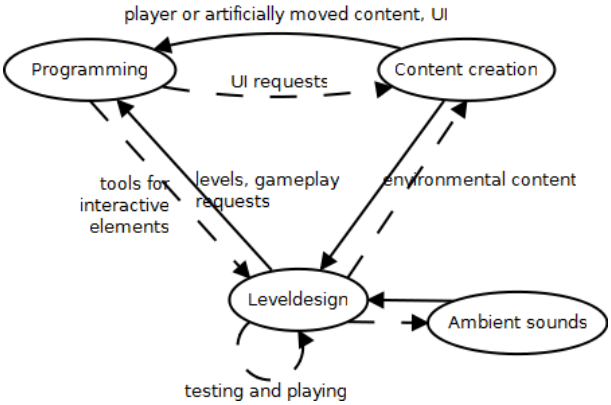


Figure 6: Level design in context to other game development tasks.

### 3.3.2 Level design in this project

The level design started right from the beginning of the project with concept plans of the game area [8] followed by modeling and texturing of the terrain. After this initial phase a level file was created and assembled in Maya, using references, so that unfinished content (e.g. low polygon models) could already be placed in the scene and be updated on the fly. In the end the whole scene was imported into Unity.

Figure 7 shows the stages of development of the level from the concept art over the terrain model to the final level rendering with buildings and other assets. The level design process took the following considerations into account:

- During the creation of the terrain, conceptual areas were planned for later object placement. (e.g. graveyard area, scenic overview, spots for buildings along the road).
- The level design was based on static geometry. To keep the player interested in exploring the level, many different floors and long paths along the terrain were created.

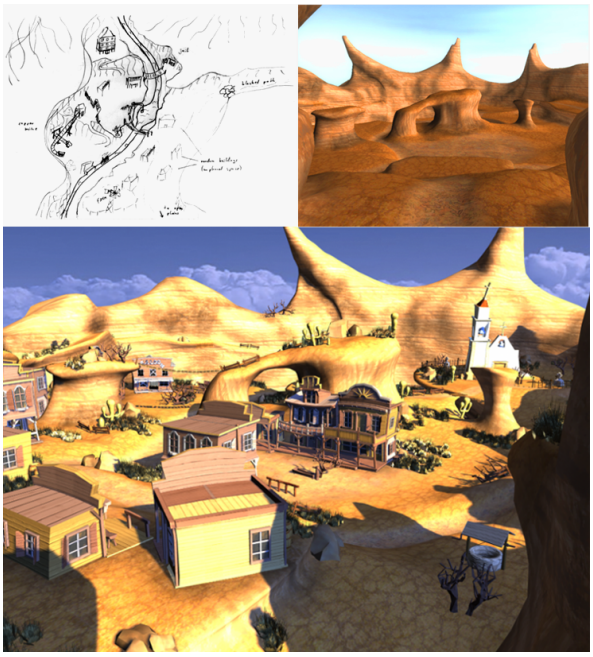

Figure 7: Three stages of level design: concept, terrain model, final scene.

- To underline the escape scenario of the game, a certain degree of hostility was added to the terrain design (e.g. in the shape of spiky rock formations).
- To keep the level simple for lighting, no caves were created.

## 3.4 Lighting

3D Production packages include powerful renderers to simulate light propagation through a scene. To this point, physically correct simulations of light are impossible because of measuring uncertainty and the inability to solve the rendering equation exactly. In addition, approaches close to physical models are often very costly and cannot be done in real-time yet.

Since computer games are real-time applications, preprocessing for global illumination is often used ('Light Baking') to achieve realistic lighting at runtime. During the process of Light Baking, the whole scene is pre-lit and the final color distribution is stored in textures. These textures can be either applied directly to a polygonal surface or saved as vertex colors. The advantage of Light Baking is that the artists can use high quality rendering algorithms to pre-calculate the lighting of a scene. Disadvantages are the added complexity in the production workflow and the lack of dynamic lights. Combining pre-calculated and real-time lighting effects can be a challenge, because numerous customized shaders and materials need to achieve acceptable color- blending effects.

#### 3.4.1 Lighting in this project

The scene was meant to be illuminated by the sunset with dominating pink and purple colors. Dark areas were avoided to keep the lighting simple. Color textures had to be desaturated, so the scene could be lit with enough power to produce a believable result. Autodesk Mental Ray [6] was used to render the scene. Its fast final gathering algorithm in combination with a light emitting sky-dome was chosen to achieve the effect of global illumination for the game's content.

The sun consisted of two simple directional light sources not affecting the sky-dome. One light-source was used for controlling the light color and intensity. The other light source was used for controlling the shapes, intensities and colors of the shadows.

It was important to know the scene's look under final lighting conditions at an early stage of the production process to define appropriate texture color guidelines. Therefore, this light setup was developed at an early stage to test different texture colors and shaders in the chosen environment setting.

The final step in lighting was light map texture baking. Diffuse color bouncing required emitting and reflecting colored objects in the scene during bake-time. These objects had to be prepared for baking. First, they were given a second UV-set, which was not overlapping or tiled.

For light-mapping and export, objects were grouped into export groups, sharing one common light-map, and were then imported separately into Unity. A building and parts of its corresponding light-map are shown in Figure 8.

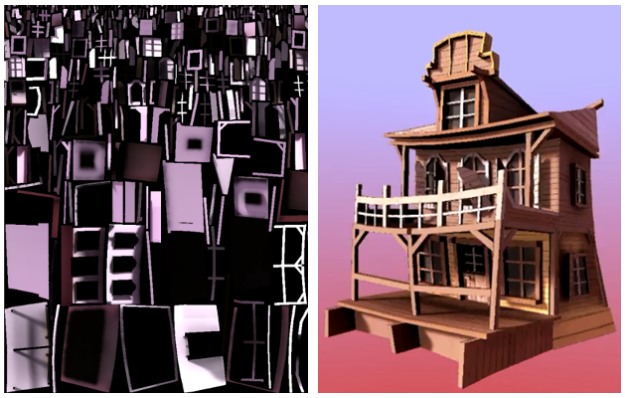

Figure 8: Building to the right and its corresponding light-map to the left.

#### 3.4.2 Concept and Presentation Rendering

In addition to the scene's lighting, rendering was also used for many different tasks during the development process. Concept renders were created in order to communicate the aesthetics of the game environment. A game trailer was produced showing some animation sequences and renders were created by artists to share their concepts and models through the team's development blog. Since rendering is a time consuming process, some additional tricks were used to improve quality in these renderings. Depth of field effects were achieved through depth-map controlled blur filters using compositing. Additionally, image post processing was used to achieve the desired quality.

### 3.5 Implementation

During the implementation phase, intensive work with the game engine started, as all the models had to be imported into Unity. Unity generally uses *.fbx files, but allows the import of *.ma files as well. These Maya files were directly imported into Unity to avoid problems with animated objects exported by Maya 2011 as *.fbx and to skip an additional step in the workflow. The files were then converted into *.fbx by Unity. This procedure makes it easy to re-import changed Maya files but it takes more time for Unity to convert all files.

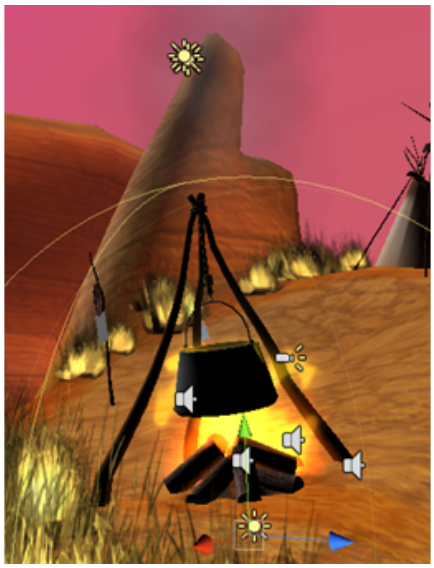

Figure 9: Fire, smoke effects and sound sources.

#### 3.5.1 Object Integration in Unity

In the first step of the object integration process the level, which had been created in Maya 2011, was imported into Unity. It was structured into different display layers, each one dedicated to a different type of models (e.g. houses, grasses, cacti). When the level was completed, the different layers were saved in separate Maya files and put together in Unity. This made it easier and faster to re-import changed layers into Unity. The different characters as well as various pickup items (e.g. dynamite) used in the game story, were directly placed into the level in Unity.

#### 3.5.2 Effects, Shader Programming

As soon as the main part of the game was finished, effects like fire and smoke (Figure 9), fog, water, dust, lens flare and explosions were implemented. All these effects, except lens flare and fog, were created with an Ellipsoid Particle Emitter or a Mesh Particle Emitter provided by Unity. Fog was simply activated in Unity's render settings.

#### 3.5.3 Sound

In the final phase background music and sound effects were added to the game. To add a sound file to an object in Unity, an 'Audio Source Component' had to be added to the specified object. In Unity's audio source options, the audio clip had to be defined and some other options like the volume were individually adapted.

### 3.6 Release

The final result of the whole production process was a game prototype with one level. Overall, it took a production team of 26 students about 1550 person hours to develop the game 'Dynamite Pete'. Table 1 gives a detailed overview of the working time spent on the individual phases of the project.

| Task | Time in person hours |
|---|---|
| Concept and references | 130 |
| Low polygon modeling | 500 |
| UV layout | 150 |
| High polygon modeling | 50 |
| Texturing | 190 |
| Rigging | 20 |
| Animation | 50 |
| Level design | 50 |
| Lighting | 60 |
| Implementation | 200 |
| Miscellaneous (presentation) | 50 |
| Organisation and feedback | 100 |
| Total | 1550 |

Table 1: Development time overview

After the end of the project, the final game was presented in a release event. The artists created a trailer video, character sheets, posters and character turntables for this event.

## 4 Further Challenges

### 4.1 Terrain Multi-Texturing

The usual approach for texturing 3D models is to apply a single texture per object. Since the terrain is a very large object, tile-able textures were used to define its surface properties. These textures are repeated seamlessly across a geometric surface representation. However, since terrain usually consists of different material, more than one tile-able texture had to be used.

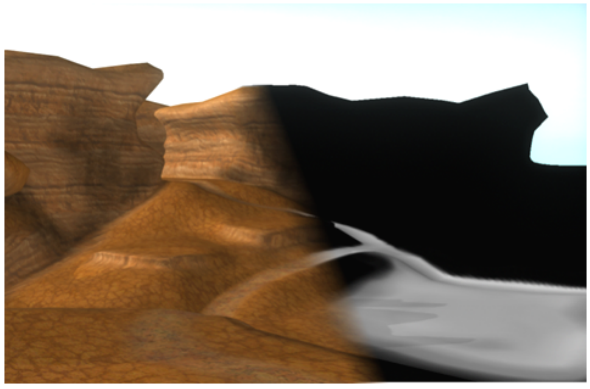

Figure 10: Left side: textured terrain. Right side: alpha map for the path layer.

Multiple layers of tile-able textures were used in combination with an alpha texture to define the global occurrence of a material. Figure 10 shows the textured terrain on the left side and the alpha map for one layer on the right side. In Maya, the material that supports this kind of texturing method is called 'Layered Texture'. In Unity, this method of texturing is only provided for terrains created from height-maps. Therefore, a custom fragment shader had to be written for the terrain in this project.

The following shows the essential part of this fragment shader code. First, the albedos of the individual overlay layers are consecutively accumulated. Finally, the calculated albedo is multiplied with the light intensity to obtain the output color.

```
//'c', 'c1', 'c2', 'c3' ... color textures
//'a' ... alpha values in rgb channels
//'lightmap' ... light map texture

//overlay 1st layer
o.Albedo = (1.0f-a.r)*c.rgb + a.r*c1.rgb;
//overlay 2nd
o.Albedo = (1.0f-a.g)*o.Albedo + a.g*c2.rgb;
//overlay 3rd
o.Albedo = (1.0f-a.b)*o.Albedo + a.b*c3.rgb;

//1.5f to control light map brightness manually
o.Albedo = o.Albedo * lightmap * 1.5f;
```

### 4.2 Concurrent working with Unity

As mentioned earlier, there are some differences between Unity Free and Unity Pro. Because the game was developed with Unity Free, a solution for the SVN problem had to be found. Unity Pro saves all asset metadata and import settings for each asset in a corresponding metafile. These files need to be versioned along with the associated asset

[2]. In Unity Free all these metadata are saved in the library directory [1]. To share a project without Unity Pro, the library directory has to be compressed and uploaded via SVN. After downloading the project from the SVN server the library directory has to be uncompressed into the Unity project folder. One negative effect of this solution is that it is not possible to work concurrently on the Unity project file. Because the whole library folder is uploaded, it is not possible to merge the different metadata, only the whole folder can be overwritten. Therefore, for multiuser projects it is recommended to use Unity Pro.

## 5 Summary

Together with our colleagues, we deployed a professional game development studio workflow in the context of a 3D content creation project. The utilization of 3D content in a 3D game put theoretical knowledge into practical use. Participating students improved not only their technical skills, but also learned valuable lessons in team work. The major challenges were different theoretical and practical understanding of the subject, unalike working methods and variable personal availability. A blog was used by most of the students to post their work and to get feedback in a constructive way. The final stages in the process of game development were rushed in about one or two weeks. At the time of the release deadline, the result was a decreased quality in game-play and lighting as well as the game mechanics, which had not been tested before. More concurrent work and earlier deadlines would have been needed to speed up the production process. Nevertheless the project was finished within the given time schedule, resulting in a nice game containing rich and detailed content.

## 6 Acknowledgements

The authors would like to thank all team members who participated in this valuable exercise and dedicated their time to this project. Thanks to Rainer Angerer, Martin Brunnhuber, Damir Dizdarevic, Brigit Faber, Markus Fellner, Roman Gurbat, Andreas Himmetzberger, Rosemarie Hochreiter, Roman Hochstöger, Bernhard Holzer, Peter Houska, Albert Kavelar, Desiree Lavaulx-Vrecourt, Andreas Lenzhofer, Thomas Mayr, Johannes Sorger, Stefan Stangl, Nicolas Swoboda, Markus Tragust and Ulrike Zauner. Our special thanks to our lecturer Markus Weilguny, who guided us through the whole production process. Last but not least we would also like to thank Associate Professor Michael Wimmer for making this lecture possible and for giving us the great opportunity to present our efforts and findings in this paper.

## References

[1] Unity 3D. Unity: Behind the scenes. http://unity3d.com/support/documentation/Manual/Behind%20the%20Scenes.html, 2010.

[2] Unity 3D. Unity: Using external version control systems with unity. http://unity3d.com/support/documentation/Manual/ExternalVersionControlSystemSupport.html, 2010.

[3] Autodesk. Autodesk maya online help: Fbik (full-body ik). http://download.autodesk.com/us/maya/2010help/index.html?url=Glossary_F_FBIK_fullbody_IK.htm, topicNumber=d0e188540, 2000-2009.

[4] Autodesk. Autodesk maya 2011: Features. http://usa.autodesk.com/maya/features, 2011.

[5] Autodesk. Autodesk mudbox 2011: Features. http://area.autodesk.com/mudbox2011/features, 2011.

[6] Autodesk. Mental ray. http://usa.autodesk.com/adsk/servlet/pc/index?siteID=123112&id=13566140, 2011.

[7] Disney/Pixar. Toy story. http://www.pixar.com/featurefilms/ts/, 1995-2011.

[8] Michael Stuart Licht. Gamasutra: An architect's perspective on level design pre-production. http://www.gamasutra.com/view/feature/2848/an_architects_perspective_on_, 2003.

[9] Lucasfilm Ltd. Star wars. the clone wars. http://www.starwars.com/theclonewars/, 2011.

[10] Unity Technologies. Unity 3d. http://unity3d.com/, 2011.

[11] Mick West. Gamasutra: Collaborate game editing. http://www.gamasutra.com/view/feature/3991/collaborative_game_editing, 2009.