Esa Pyykölä

Creating a chat room using Unity Game Development Tool

Opinnäytetyö
KESKI-POHJANMAAN AMMATTIKORKEAKOULU
Tietotekniikan koulutusohjelma
Huhtikuu 2011

Yksikkö	Aika	Tekijä/tekijät
Ylivieska	Huhtikuu 2011	Esa Pyykölä

Koulutusohjelma Tietotekniikka, insinööri
Työn nimi Creating a chat room using Unity Game Development Tool

Työn ohjaaja	Sivumäärä
Hannu Puomio	35

Työelämäohjaaja -

Tahdoin tutustua Unity Game Development Toolin tarjoamiin Internet-mahdollisuuksiin, joten tein keskusteluhuoneen käyttäen Unityä itseään. Tämä kaikki oli mielekkäämpää kun oli jokin päämäärä, joka tässä tapauksessa oli keskusteluhuone. Oman projektini lisäksi työni sisältää tietoa Unitystä.

Lisäksi halusin välttää käyttämästä kolmannen osapuolen serveriohjelmistoja ja toteuttaa kaiken Unityn itsessän tarjoamilla työkaluilla. Tämän lisäksi ohjelman piti pysyä mahdollisimman pienenä, joten se ei veisi niin paljon tilaa muistitikulta. Ohjelmalla voi liittyä tai luoda keskusteluhuoneen, joten dedikoitua palvelinta ei tarvitse.

Toisessa kappaleessa käyn läpi Unity-kehitysympäristön historiaa, ominaisuuksia ja työkaluja. Kolmannessa esittelen tärkeimmät Unity:n verkkofunktiosta. Neljäs kappale koostuu tietoliikennetermeistä. Viides ja viimeinen kappale sisältää kertomukseni keskusteluhuone-projektistani.

Asiasanat
Chat room, game programming, keskusteluhuone, networking, peliohjelmointi, Unity

CENTRAL OSTROBOTHNIA UNIVERSITY OF APPLIED SCIENCES	**Date** April 2011	**Author** Esa Pyykölä
Degree programme Information Technology Programme		
Name of thesis Creating a chat room using Unity Game Development Tool		
Instructor Hannu Puomio		**Pages** 35
Supervisor		

After working on Unity for nearly half a year the interest in its networking capabilities and the possibilities of making multiplayer games and/or applications grew. Therefore creating a chat room using Unity was an excellent opportunity to create something useful while learning the basics of Unity multiplayer programming.

The first requirement was to avoid using third party servers and focus solely on Unity's own networking functions and processes. The second requirement was to keep the program as small as possible, so that it could be taken on a USB flash drive and used where ever one wanted to. The third requirement was to make it so that a player can create a server or connect as a client with the same software.

In the second chapter the reader shall be given information about Unity Technologies' Unity game development tool. It will include Unity's history, its technological abilities and its various tools. In the third chapter three of Unity's networking functions are introduced. In the fourth chapter there is some information on data communications to help with the terms used and an introduction to a chat room service. The fifth chapter focuses on the development steps of my project including planning, development and further development plans. The sixth and the seventh chapters consist of the additional appendixes and the sources used.

Key words
Chat room, game programming, keskusteluhuone, networking, peliohjelmointi, Unity

Terms and Abbreviations

NAT:	Network Address Translation, used for modifying the information datagram packet headers
Server:	A computer program that services clients or a computer running multiple such services
Client:	An application or system that connects to a server to access a remote service
RPC:	Remote Procedure Call, permits calling a function or a procedure on another computer
Fillrate:	Tells how many pixels a graphics card can render and write to video memory in one second

TABLE OF CONTENTS

1 INTRODUCTION

After working on Unity for nearly half a year the interest in its networking capabilities and the possibilities of making multiplayer games and/or applications grew. Writing this thesis on a subject which was familiar and interesting served as a significant motivator. That led to an idea to create a chat lobby where one can invite people to play games and chat with other players.

The first requirement was to avoid using third party servers and focus solely on Unity's own networking functions and processes. The second requirement was to keep the program as small as possible, so that it could be taken on a USB flash drive and used where ever one wanted to. The third requirement was to make it so that a player can create a server or connect as a client with the same software. This way people can make their own servers on the fly and invite the people they want e.g. for a quick game for the duration of a lunch break. On the other hand, it is possible to use a dedicated computer to run the server around the clock to provide a proper server.

This thesis includes the basics for creating a multiplayer software with Unity. The thesis focused on the chat lobby alone. It does not include the creation of a game for this chat room service as that would have doubled or tripled the software development duration. This chat room service does not include support for third party servers because that is not the first step for me to take when exploring the networking capabilities for Unity.

Having no experience in Unity networking, everything had to be learned from the scratch. The following things had to be learned: How to make a server and connect to it, how to list players while keeping server security by separating client and server information, how to create the GUI so that it would support chatting and listing players, and how to invite a player for a game and send messages between clients.

Unity Technologies' own site was used as a source as they have the most up-to-date information on their corporation and technologies. A good amount of information was found on their main page and reference manual. For other subjects English Wikipedia was used. Wikipedia was chosen because there is up-to-date basic information on the subjects and crowding this thesis with technical specifications and standards was not wanted.

In the second chapter the reader shall be given information about Unity Technologies’ Unity game development tool. It will include Unity’s history, its technological abilities and its various tools. In the third chapter three of Unity’s networking functions are introduced. In the fourth chapter there is some information on data communications to help with the terms used and an introduction to a chat room service. The fifth chapter focuses on the development steps of my project including planning, development and further development plans. The sixth and the seventh chapters consist of the additional appendixes and the sources used.

2 UNITY GAME DEVELOPMENT TOOL

2.1 History of Unity

The development of Unity started in 2001 and the first version was launched at Apple's WWDC(Worldwide Developers Conference) in 2005. During the 2006 Unity was named a runner-up in the Apple Design Awards for ”Best use of Mac OS X Graphics”. This occasion was the first time a game development tool received a merit of high standing. (Unity home page 2011.)

In 2007 Unity 2.0 was released and the first Annual Unite Developer Conference was held by Unity Technologies. During the year 2008 Unity Technologies had tripled in size. Along with this feat Unity for iPhones was released and Unity Technologies became an authorized middleware provider for the Nintendo Wii and, also Cartoon Network released a game called FusionFall which has been played over 8 million times. Overall the year 2008 was a good one for Unity Technologies as their product was a finalist in Game Developer magazine's Front Line award. (Unity home page 2011.)

In 2009 the big change came and Unity Technologies released a free version of Unity at Unity Technologies' Developers Conference. Again this year Unity Technologies tripled in size and received a 5.5 million dollar investment. Award wise 2009 was a good year for Unity Technologies. They were named as ”One of the Top Five Game Companies of the Year” by game developers ' website Gamasutra. Develop Magazine credited them starting a ”Tech Revolution”. Again in 2009 they were a finalist for the Front Line Award by Game Developer magazine. To top it all Unity Technologies' CTO Joachim Ante was recognized with an individual award of making it into Develop Magazines ”30 under 30” list. (Unity home page 2011.)

In February 2010 Unity had reached 100,000 developers and the next version of Unity was previewed in March at Game Developers Conference(GDC). A major technical advancement was made in Web player development for Google's Chrome browser as it no longer requires Unity Web player to be installed. Later that year a big game corporation Electronic Arts announced a deep partnership with Unity Technologies. During the fourth fiscal quarter the latest version of Unity was released which included many new technologies and the popularity of Unity was growing exponentially with Unity developers hitting 250,000 by October and Unity Web Player was installed over 35 million times. An Asset Store was released for Unity at Unity Technologies’

Developers Conference(Unite) in November which allows Unity developers to sell their assets to other Unity developers. During this conference a new unit called Union in Unity Technologies was created which helps Unity games reach broader audiences.(Seyler 2010.) (Unity home page 2011.)

In 2010 Unity Technologies won many prizes and recognitions. To mention a few they won the Wall Street Journal's ”Technology Innovation Award”, Develop Magazine's ”Grand Prix Award” and ”Technical Innovation Award” and yet again they were finalists in Game Developer magazines Front Line Award. (Unity home page 2011.)

Currently Unity has many major companies as registered users such as Cartoon Network, Disney, Microsoft, Coca-Cola, NASA and LEGO. Alongside with these are game studios of varying sizes and individual developers. (Unity home page 2011.)

2.2 Licensing

Unity focuses on creating a technology that is usable and powerful and making it as easy to use as possible. Unity features a large array of graphics technologies from real-time lightning rendering, built-in physics to AAA-title light mapping and occlusion culling. Currently Unity supports most platforms that are in wide use such as PC, Mac and Web. In addition to computers Unity supports handheld devices like Apple's iPhone and iPad and mobile phones using Google's Android. Unity also supports the seventh generation consoles Xbox 360 and Nintendo Wii with Playstation 3 support coming in the future. (Unity home page 2011.)

There are many different licenses available for Unity. The free version of Unity contains only PC/MAC and Web Player deployment and lacks the most advanced graphics technologies such as Depth of Field. Unity Pro features the full arsenal of graphics technologies and adds a debugging profiler which allows the developer to debug the software more efficiently. Unity Pro is required for iOS Pro, Android Pro and Asset Server licenses. (Unity home page 2011.)

There is a standard version of iOS license which is stripped of some technologies such as video playback/streaming and multiplayer assets compared to iOS Pro. For Google's Android there is only the Pro version but standard Android license has been announced. (Unity home page 2011.)

Other licenses such as educational or game console licenses have to be negotiated with Unity sales representatives. (Unity home page 2011.)

2.3 Rendering

Rendering in Unity 3.1 supports Deferred Rendering which allows a large number of dynamic lights even in browser based games. The advantages of Deferred Rendering are that there are no limits to how many lights can affect an object. All lights can have cookies and shadows and they can interact with normal maps, making objects look even better. As lightning cost in this type of rendering is proportional to the size of the light, small lights are cheap performance wise. There are disadvantages however. These include no real anti-aliasing or semi-transparent object support and lightning model support is limited which means one cannot have drastically different lightning models on different objects. Deferred Rendering requires Unity Pro and a graphics card that supports Shader Model 3.0 or higher. It is not supported on mobile platforms yet. (Unity home page 2011.)

Unity also has a plentiful number of built-in shaders from simple Diffuse to Self Illuminated Bumped Specular. There are five categories of built-in shaders: Normal, Transparent, Transparent Cutout, Self-Illuminated and Reflective Shader categories. Normal shaders are the most basic and unspecialized. They are most useful for nonreflective materials such as wood or plastics. (Unity home page 2011.)

Transparent shaders are meant for transparent or semi-transparent objects. These shaders use the alpha channel of the base texture which controls the object’s transparency level. This shader is excellent for glass surfaces and HUD interfaces. (Unity home page 2011.)

Transparent Cutout shaders are meant for objects that have either fully opaque and fully transparent parts. It does not support partial transparency. Objects that are ideal for this shader type include trees, grass and chain fences. (Unity home page 2011.)

Self-Illuminated shaders, as the name suggests, illuminate themselves. This means they do not need outside lights to shine upon them. These other lights add to the lightning level of the objects. This shader type is mostly used on light emitting objects such as lights, displays or objects that should be visible regardless of present lightning e.g power-ups or collectibles such as coins. (Unity home page 2011.)

Reflective shaders are meant for e.g. chrome, liquid surfaces or video monitors. The number of reflections are based on the alpha channel of the base texture. (Unity home page 2011.)

Unity offers great scalability for high level shaders so that they run well even on lower end hardware. This is achieved by extensive use of fall backs in the shaders. This allows everyone to have the best possible experience depending on their hardware. This is further helped by graphics emulation by Unity editor during development to simplify testing. (Unity home page 2011.)

The latest version of Unity, Unity 3.0, includes Surface Shaders which simplifies the process of creating shaders for multiple platforms. It does this by giving an option to write a shader code in Cg / HLSL instead of low level vertex/pixel shader programs. (Unity home page 2011.)

Performance wise Unity offers many technologies to improve rendering efficiency. Unity uses Batching to minimize draw calls by automatically combining geometry into batches and wields a custom made GLSL(OpenGL Shading Language) Optimizer which was developed for Unity and it doubles or even triples the fillrate. Perhaps the most significant performance boost is Occlusion Culling which was developed for Unity with Umbra Software. It is used to render only those objects that are visible to the camera and thus needed for example hundreds of crates are not drawn if they are behind a wall. Unity also includes Frustrum Culling which restricts object rendering to those objects which are visible in the view angle of the camera, for example objects behind the camera are not drawn. (Unity home page 2011.)

2.4 Lightning

Unity has the support for realtime shadows. This means objects can cast shadows onto other objects and the object itself. This greatly enhances the visual quality of a game but also comes with a cost. They are very expensive to render because the process is two phased. First the potential shadow casters need to be rendered into the shadow map and then all shadow receivers are rendered with the shadow map. Due to this it is recommended that one does not have too many realtime shadows and prefer non-realtime where it is possible. (Unity home page 2011.)

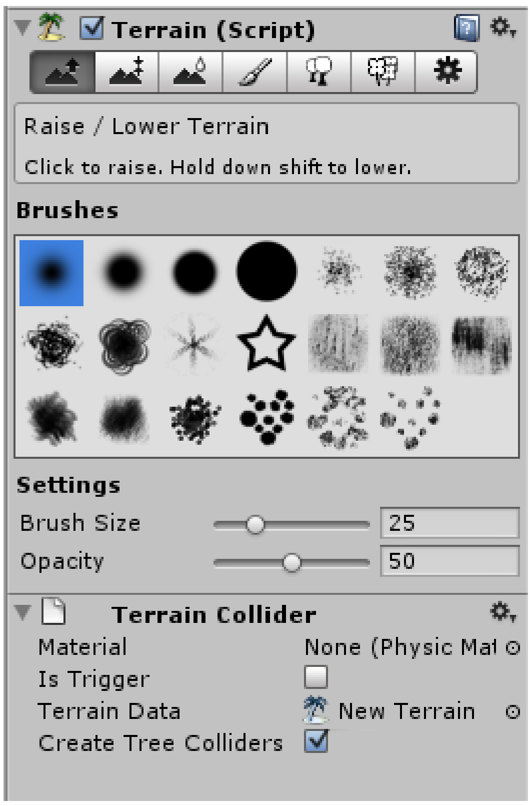
Unity Pro includes a Screen Space Ambient Occlusion(SSAO) image effect, which makes the game look much better by darkening creases, holes and surfaces which are in close proximity. As realtime shadows, this image effect is expensive to calculate although Unity 3.0's Deferred Lightning reduces the performance drop from this effect. The performance of this image effect is dependent on screen resolution and given SSAO parameters. (Unity home page 2011.)

In addition to SSAO Unity contains over twenty other image effect filters which are included in Unity Pro license. To mention a few there is blur, bloom, color corrections, fish eye, sun shafts and noise effects. These all can give a game unique effects and visual styles, e.g. using motion blurring in racing games to enhance the impression of speed or bloom to have shining surfaces. Noise on the other hand can be used for example in horror games to create a more intense atmosphere. (Unity home page 2011.)

Perhaps the most effective of Unity's lightning repertoire is the ability to create lightmaps. As Unity 3.0 supports creating lightmaps, developers do not have to do this inside 3d modeling software. This allows developers to create their levels inside Unity from blocks if necessary. Lightmapping in Unity is provided by Illuminate Labs' Beast. Unity's lightmapping supports dual lightmapping which means one lightmap is used for distant objects and the other one contains only bounce light. This makes the lightmap look much better as it integrates the main character well in environments, for example.
(Unity home page 2011.)

2.5 Terrains

Unity offers an excellent tool for creating terrains which are the base for most games. When one creates a terrain it is a flat piece of land. Then one starts sculpting it with the tools Unity provides. To roughly modify the terrain one uses raise and lower terrain tools. These either increase the elevation of the land or decrease it inside the brush area. After using either of these tools the terrain looks rough. To finish up the terrain one uses the smoothening tool which smoothens the terrain inside the brush. (Unity home page 2011.)


Terrain (Script)

Raise / Lower Terrain

Click to raise. Hold down shift to lower.

Brushes

Settings

Brush Size	25
Opacity	50

Terrain Collider

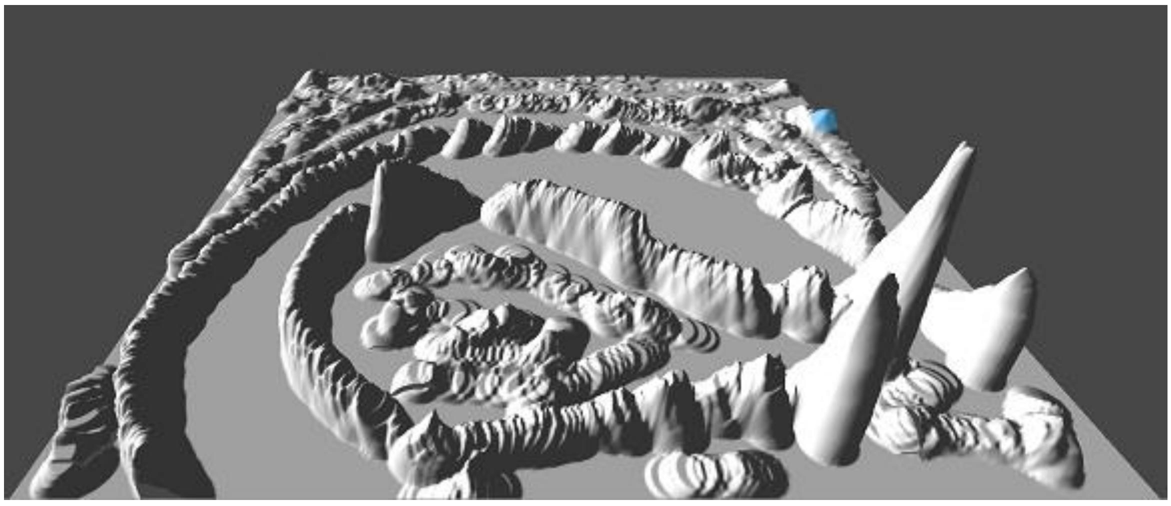
Material	None (Physic Mat
Is Trigger	☐
Terrain Data	New Terrain
Create Tree Colliders	✓

PICTURE 1. Terrain tools. At the Top there are the different tools: Raise/lower terrain, set terrain height, smooth, paint texture, plant trees, plant grass and rocks and options.

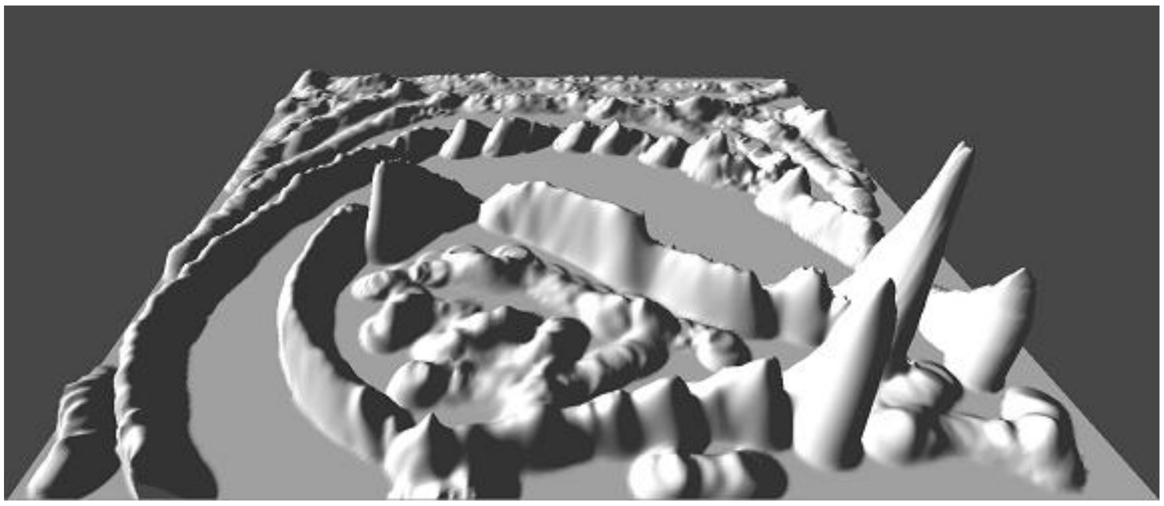
In addition to creating just the land mass for one's games Unity has the tools for creating the vegetation including trees, bushes and grass. One can paint these on the terrain with a variety of different brushes just like when modifying the terrain itself. Unity optimizes these trees and bushes when they are far away by making them billboards which look like 3d objects although in reality they are 2d sprites. (Unity home page 2011.)

A new feature in Unity 3.0 was the ability to create trees with the tree generator. The tree generator allows the developer to create trees with many possible options. This tool is excellent for creating lush forests or jungles with many different looking trees. The trees created with this tool can be used either as standard game objects or integrated into the terrain engine. (Unity home page 2011.)

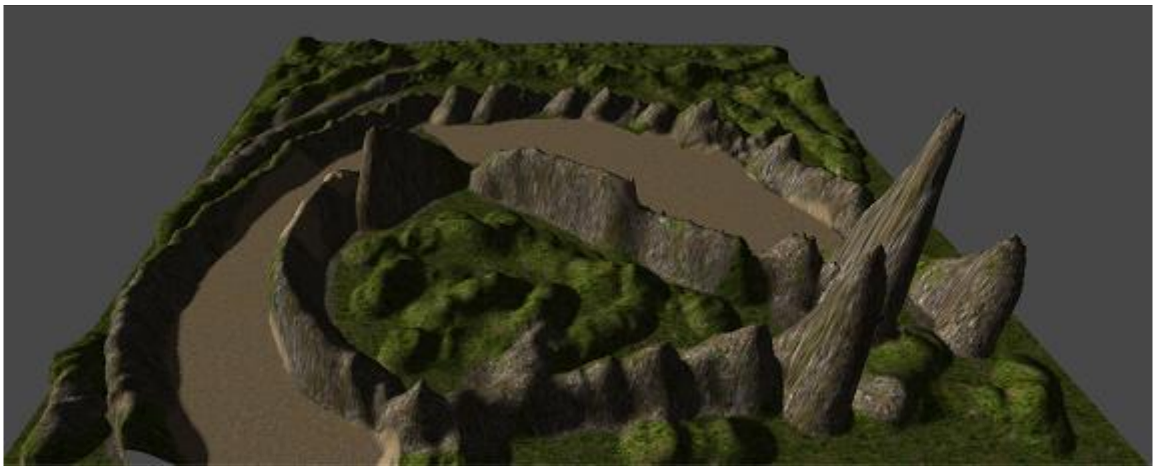
PICTURE 2. Fresh terrain. This terrain was just created and is ready for editing.



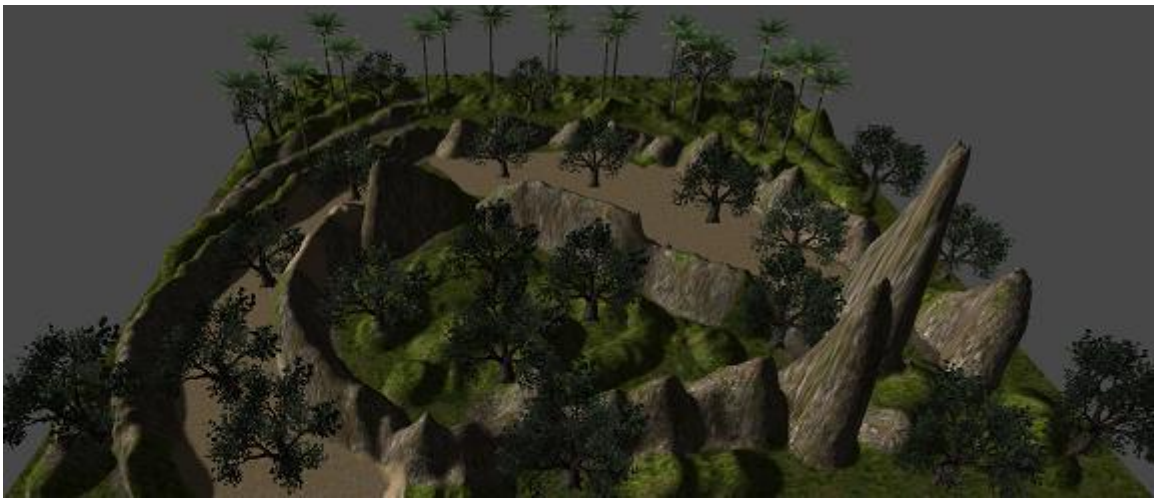
PICTURE 3. Raise/Lower terrain. Thirty seconds of using raise terrain tool and the rough terrain is ready.



PICTURE 4. Smooth tool. The quick use of smooth tool gives a less volcanic environment with eroded hill sides.



PICTURE 5. Paint texture. The quick use of the texture tool and the landscape looks much better.



PICTURE 6. Paint trees. Now our landscape is complete with vegetation.

This whole process of creating a terrain from a flat piece of plane to a textured and flora bearing landscape took a few minutes. This is an excellent way of demonstrating the power of Unity's terrain editor.

2.6 Physics

Unity’s physics engine supports rigidbodies, which means one's game objects can act under the control of physics. This means a game object can be given values in e.g. torque or forces and they will act correspondingly. Alongside with rigidbodies Unity supports a variety of joints like hinges, springs and ball-sockets. These enable the developer to create doors, chains or pendulums at ease. Physics in Unity is handled by NVIDIA’s PhysX. (Unity home page 2011.)

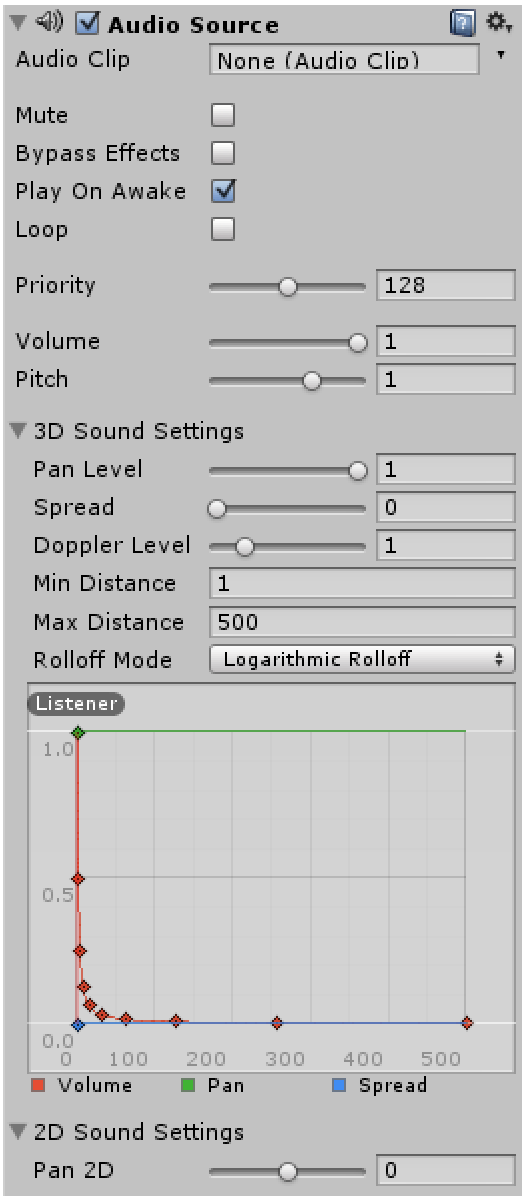
In addition Unity also has support for ragdolls, soft bodies, cars and cloth simulation. Ragdolls in Unity can be created using the Unity's own ragdoll wizard. To create a ragdoll simply import a 3D package file which includes skinned meshes and open up the wizard to assign body parts. (Unity home page 2011.)

The soft bodies feature allows the creation of e.g. semi-deflated beach balls that can interact with the environment and the cloth feature allows the creation of physically simulated cloth that interacts with the environment. With the cloth feature it is possible to optimize the clothing the game characters wear and make them physically accurate. (Unity home page 2011.)

Finally Unity supports wheel physics which makes it easier to create cars. This is achieved by using a Wheel Collider. It is a special collider designed for ground vehicles. It includes collision detection, wheel physics and a slip-based tire friction model. (Unity home page 2011.)

2.7 Audio

There are three types of audio objects in Unity, audio sources, audio listeners and audio reverb zones. Listeners are just objects that receive sound and have no options other than on or off. Reverb zones are used to gradually distort sound. Audio sources have many options to edit or filter the sound they send. Editing sound is very simple since all key audio properties are operated by dragging sliders or changing attenuation curves. (Unity home page 2011.)



Audio Source

Audio Clip	None (Audio Clip)
Mute	☐
Bypass Effects	☐
Play On Awake	✓
Loop	☐
Priority	128
Volume	1
Pitch	1

3D Sound Settings

Pan Level	1
Spread	0
Doppler Level	1
Min Distance	1
Max Distance	500
Rolloff Mode	Logarithmic Rolloff

2D Sound Settings

Pan 2D	0

PICTURE 7. Options for an audio source.

It is possible to edit standard things such as sound pitch, whether the sound should loop and, what the priority of the sound is if sound channels are limited and volume. Furthermore, there are 3D sound settings in which one can edit pan level, spread, Doppler level and distances, and set roll off attenuation curve. In 2d sound settings there is only a panning setting. (Unity home page 2011.)

In Unity Pro one can use built-in audio filters which include low pass, high pass, echo filter, distortion filter, reverb and chorus. These filters are used by applying them into AudioSource or AudioListener objects. Audio in Unity is provided by FMOD which is a proprietary audio library from Firelight Technologies. The following formats are supported: MP3, Ogg Vorbis .ogg, AIFF, WAV, MOD, IT, S3M and XM. (Unity home page 2011.)

2.8 Programming

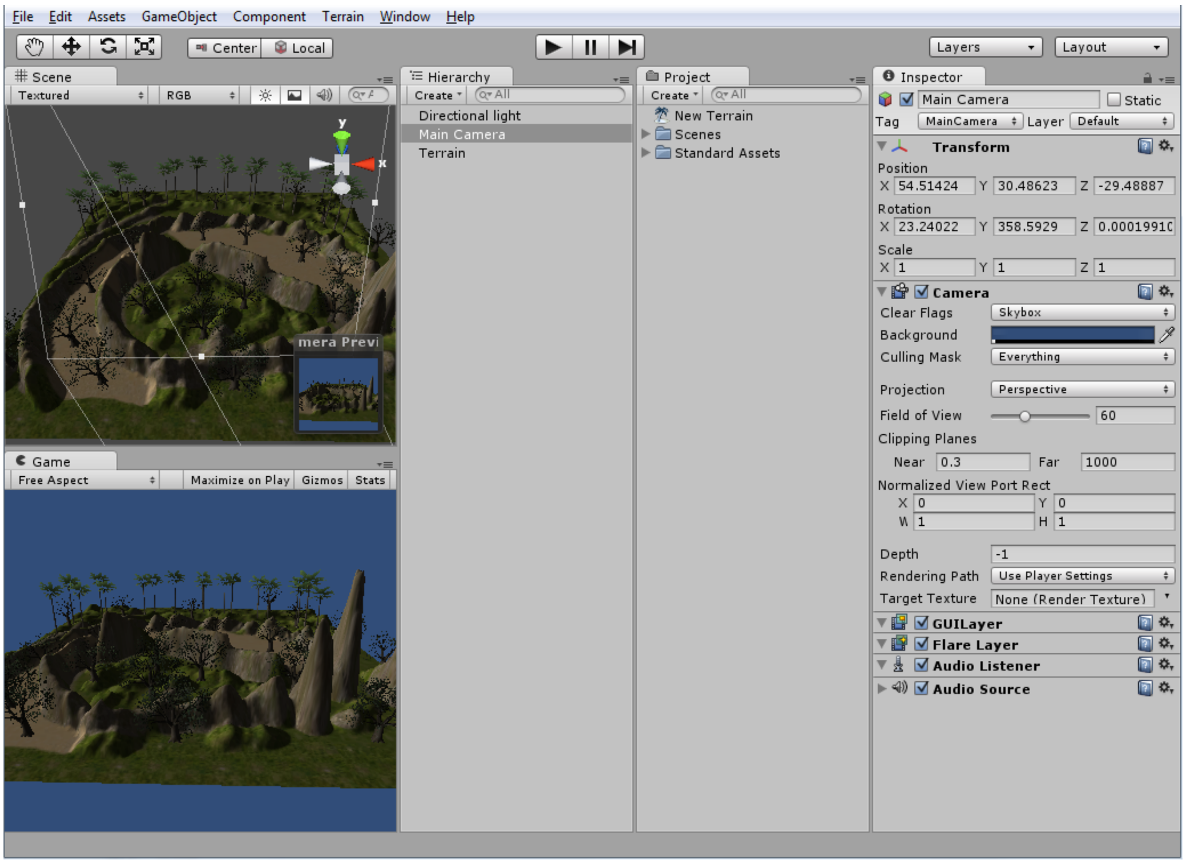
Unity supports three different scripting languages: JavaScript, C# and Boo, which is a dialect of Python. All these languages support .NET libraries which allows the use of databases, regular expressions, XML, networking etc.. The performance of these languages in Unity is nearly as good as C++, and JavaScript implementation is as fast as C# and Boo. All this is run on Mono platform which is an Open Source .NET platform. (Unity home page 2011.)

Unity supports a great deal of visual properties as all variables defined public in script are shown in an inspector window. This way one can edit variables by simply using one's mouse to drag them. This combined to the fact that many operations in Unity require only single line of code, such as rotating or moving an object. The possibility of referencing objects directly or by creating tags or even by proximity or touch makes Unity very flexible and easy to use. (Unity home page 2011.)

Unity has full MonoDevelop integration which means one can use an IDE which works completely on Unity. This allows one to sync one's project with MonoDevelop to gain auto-completion. Using MonoDevelop one can debug one's Unity scripts as any other software by creating break points, single stepping line by line and inspecting values. Unity does support Visual Studio so it is possible to sync a project to it. (Unity home page 2011.)

Last but not least of Unity's programming tools there is the profiler. It allows one to see performance data visually and in real-time. Shown statistics include CPU usage, rendering and memory. Under these main categories the toll each process takes from these resources is shown. For example CPU usage has the following processes: rendering, scripts, garbage collector, physics and others. The graph is drawn in real-time and by pausing the game one can inspect which processes cause performance hiccups or any unwanted events. This speeds up the debugging and final stages of the development. (Unity home page 2011.)

2.9 Editor



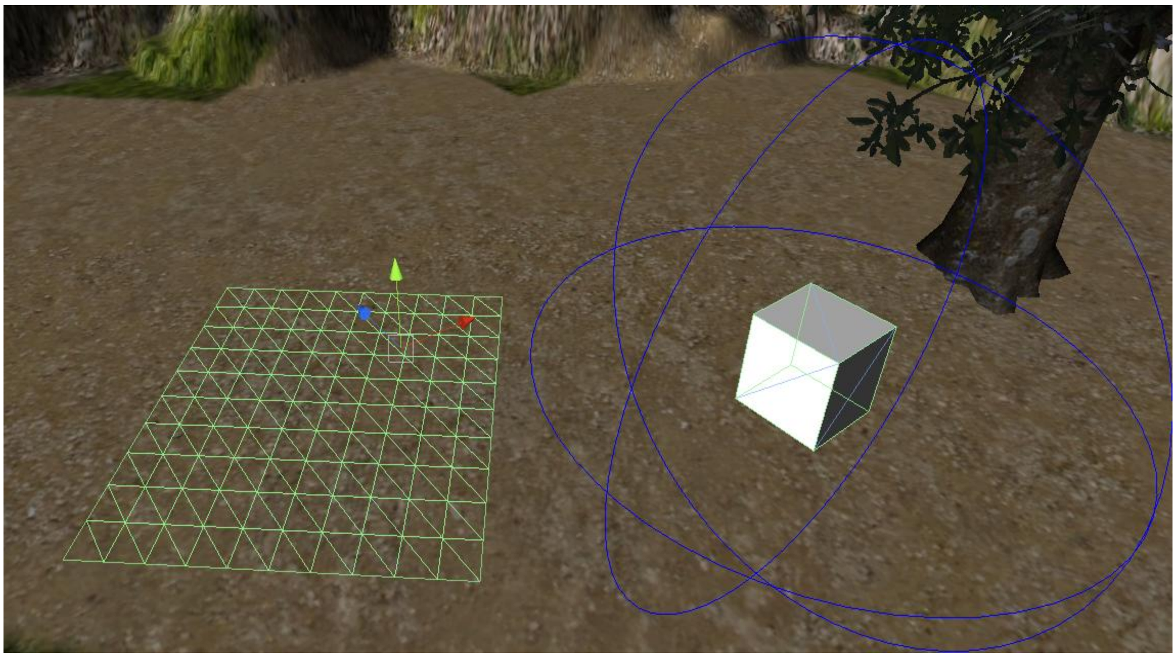
Hierarchy: Directional light, Main Camera, Terrain

Project: New Terrain, Scenes, Standard Assets

```
Main Camera                         Static
Tag  MainCamera   Layer  Default
Transform
Position   X 54.51424  Y 30.48623  Z -29.48887
Rotation   X 23.24022  Y 358.5929  Z 0.0001991C
Scale      X 1         Y 1         Z 1
Camera
Clear Flags        Skybox
Background
Culling Mask       Everything
Projection         Perspective
Field of View      60
Clipping Planes
  Near 0.3         Far 1000
Normalized View Port Rect
  X 0              Y 0
  W 1              H 1
Depth              -1
Rendering Path     Use Player Settings
Target Texture     None (Render Texture)
GUILayer
Flare Layer
Audio Listener
Audio Source
```

PICTURE 8. An editor view showing all the basic Unity editor features. The scene window is where the objects are placed. The game window shows the game view of the scene. The hierarchy window shows all the game objects in the scene. The project shows all the assets in the project. The inspector shows the details of a GameObject such as transform, attached components and scripts.

The editor itself in Unity is one of the reasons which makes it so easy and fast to use. Since the editor practically is a world builder, it is possible to just drag and drop one's assets into the scene instead of doing it in code, and all assets can be previewed from 3d models and textures to audio files. Unity has many built-in features that make building a scene much easier. These include surface and grid snapping which allows positioning objects very quickly and accurately. Then there are triggers and gizmos. Certain events are triggered when entering a designated trigger area. These are very useful when creating e.g. automatic doors or cut scenes. All triggers are visualized in the editor and can be edited like any other objects by rotating, scaling and moving them around. Gizmos on the other hand are visual aids used for example in debugging. A gizmo can be a simple line or a sphere to a preview of the game camera or occlusion culling preview. Gizmos do not show in the game window nor in the built version of the game. (Unity home page 2011.)



PICTURE 9. A trigger and a gizmo. On the left there is a trigger and on the right there is a wire sphere gizmo.

When the building blocks are in place it is time to test, and in Unity that means pressing the play button and nothing else. Pressing the play button makes the game run and one can test it then and there. It functions in the same way as a deployed version and it informs of any errors or warning in the debugging log in real-time. During the play testing it is possible to pause the game and go through different variables and the positions of objects. (Unity home page 2011.)

Unity's editor offers unlimited customization as it is possible to create one's own tools in it. This means one can create tools to rig the AI or manage cut scenes. These custom windows with their functionality are created inside Unity by referencing EditorWindows instead of MonoBehavior. (Unity home page 2011.)

Assets, be they scripts, 3d models or textures, can be imported in almost any format as Unity supports a wide range of applications and formats. For example 3d models can be imported into Unity from major commercial products like Maya or 3D Studio Max to freeware Blender. (Unity home page 2011.)

In Unity it is possible to edit an asset in another software and just press save and it will be updated in one's project. This means it is not necessary to manually import one's assets every time they are modified. This easy to use approach is visible in the handling of textures as it is possible to save multi-layered Photoshop files and Unity automatically compresses them. In addition it is possible to convert any texture to a normal-map which is done automatically, and Mipmap generation is supported in three different methods: Detail Fade, Kaiser Filters and

Gamma Correction. In addition to textures Unity can handle TrueType fonts. (Unity home page 2011.)

Supported 3D packages are: Full support: Maya(.mb/.ma), 3D Studio Max(.max), Cheetah 3D(.jas), Cinema 4D(.c4d), Blender(.blend), Carrara, COLLADA, Lightwave, Autodesk FBX(.dae) and XSI 5.x. Partial support: SketchUp Pro, Wings 3D, 3D Studio(.3ds), Wavefront(.obj), Drawing Interchange Files(.dxf). (Unity home page 2011.)

For advanced users there is a possibility of post-processing incoming assets using C#. This allows developers to shape Unity's importing to their wanted aspects. (Unity home page 2011.)

For larger projects Unity offers an Asset Server which is a version control system. All the actions, updates, commits etc., are done inside the Unity editor. Asset server is based on PostegreSQL which is famous for its reliability, data integrity, easy administration and backups. Asset server is supported on all three platforms: Microsoft Windows, Mac OS X and Linux. (Unity home page 2011.)

The latest addition into the editor is the Asset Store. There developers can sell their assets or buy from other developers just as in an application store. Asset store contains everything from tutorials and guides to actual assets such as textures and scripts all the way to complete modules. (Unity home page 2011.)

2.10 Networking

Unity has plenty of easily available networking functionalities. These include WWW interfaces, a possibility to include javascript or AJAX, support for .NET socket libraries and third party software which allows the creation of MMOs(Massively Multiplayer Online). (Unity home page 2011.)

The WWW interface in Unity allows access to web pages and web services. This is done simply in one function call which starts the downloading from the designated website. WWW-function can be used for example in retrieving high scores or textures. (Unity home page 2011.)

If the software is deployed in a web player format it can communicate seamlessly with javascript and AJAX aspects of the container web page. This means the web page can call functions inside the web player content and the web player content can call functions inside the web page. This can be used for example in preventing deep linking. (Unity home page 2011.)

Unity supports .NET socket libraries for real-time networking by opening TCP/IP sockets or sending UDP messages. These libraries make speaking XML and connecting to databases typically easy for Unity. (Unity home page 2011.)

Unity itself does not come with the functionality for creating massively multiplayer games but other companies have come up with solutions for creating them. These include Electrotank Universe Platform, which is used for many types of games from social to action games. Photon Socket Server which is used in a couple of Unity's major games like Paradise Paintball and Smartfox Server which is used in several Unity games. (Unity home page 2011.)

2.10.1 Master Server

Master server is a separate service which is used to list servers and make them available to a broader audience. With Master Server it is possible to connect to games if they are unavailable due to firewall or NAT. In addition a Master Server also hides the player's IP address and port details. To make a game server available to Master Server it must be registered. To register, the server must provide a GameType, GameName and a comment. GameType is the defining field of a server. Games with the same GameType are listed as possible servers to connect, thus they must not be named Game1 or AwesomeGame. Selecting a GameType like e.g. SuperShooterV2.1.12b which contains the game's name and its version will be unique and a Master Server knows which games to pool together. Version number blocks earlier versions of SuperShooter from connecting. (Unity home page 2011.)

GameName is the name of the server which is shown when a server list is requested. By default comment row is self-explanatory but if a developer is to modify master server source code it can be implemented e.g. as a password field. Unity Technologies has a dedicated Master Server which everyone can use but the source code of Master Server and the Facilitator is available for free as ready to build project files. Facilitator is used for NAT punch-through and is a separate part of the Master Server. (Unity home page 2011.)

3 UNITY INTERNET FUNCTIONS

3.1 NetworkView

A network view is a gateway to creating multiplayer games in Unity. Network views are very easy and simple to use but have lots of functionality hidden in them. A network view has the following properties: State Synchronization, Observed and View ID. (Unity home page 2011.)

State Synchronization has three options: Off, Reliable Delta Compressed and Unreliable. Setting state synchronization off is best used if only RPCs are used. Reliable Delta Compressed mode only sends the observed data if it changes. If any packets are lost they are sent again automatically. This mode is ordered which means if a packet is lost in the middle it will be resent and all later packets are queued after it. The Unreliable options sends the state at all times and thus uses more bandwidth but packet loss impact is minimized. (Unity home page 2011.)

Observed property holds the component which is monitored. This can be an animation, transform, rigidbody or a script. The status of the monitored component is sent across the network. If the software uses only RPC calls then this property can be left empty. If the observed object is a script, the data must be explicitly serialized in the script. This is done with the OnSerializeNetworkView-function. (Unity home page 2011.)

View ID is used to identify different NetworkViews. It has two properties: Scene ID and Type. Scene ID identifies the NetworkView in that scene and Type defines if the NetworkView is saved to a scene or if it is allocated at runtime. (Unity home page 2011.)

3.2 RPC – Remote Procedure Call

An RPC(Remote Procedure Call) allows one to call a function on a remote machine. It does not differ much from calling a normal function but there are some differences. The number of parameters given to an RPC is practically unlimited but more parameters mean more used bandwidth. Another difference is that it must be decided who receives that function call through an RPC. It can be called on all, everyone else but the caller, only the server or a specific player.

(Unity home page 2011.)

Uses for an RPC could be for example a client calling other players that it has received an item or a server starts the game only after a specific number of people have joined the game. RPCs can be buffered and this allows an easy way of synchronizing a new player to a game. As the RPC calls are buffered a new player joins the game and receives all the previous calls in order and is then at the same level as the other players. (Unity home page 2011.)

RPC function is created by inserting a “@RPC” line before a normal function, this allows it to be called as RPC. RPCs can take the following parameters in Unity: integers, floats, strings, NetworkPlayers, NetworkViewIDs, Vector3s and Quaternions. An RPC call looks like this: networkView.RPC(“MyAwesomeFunction”, RPCMode.Server, “Hello World!”);. First the function name is given, then to whom it is called and last the parameter is given. The function that RPC would invoke looks like this:

```
@RPC

function MyAwesomeFunction(incomingParameter : String)
{
Print(incomingParameter);
}
```

(Unity home page 2011.)

In this example the function is only called on the server and the server would print “Hello World!” on its console. In the function itself it is possible to take extra information about the sender of the RPC by a NetworkMessageInfo struct. Through this struct it is possible to get a timestamp, sender and networkView information of the sender. It is added into the function like this:

```
@RPC
function MyAwesomeFunction(incomingParameter : String, senderInformation :
NetworkMessageInfo)
{
Print(IncomingParameter);
}
```

(Unity home page 2011.)

3.3 OnSerializeNetworkView

This function is used to customize the synchronization of variables in a script which has a network view attached to it. It automatically detects if the serialized variable should be sent or received. An example of an OnSerializeNetworkView from Unity script reference:

```
var currentHealth : int = 0;

function OnSerializeNetworkView(stream : BitStream, info : NetworkMessageInfo) {
        if (stream.isWriting) {
                var healthC : int = currentHealth;
                stream.Serialize(healthC);
        } else {
                var healthZ : int = 0;
                stream.Serialize(healthZ);
                currentHealth = healthZ;
        }
}
```

(Unity home page 2011.)

4 DATA COMMUNICATIONS

4.1 Server

A server is a computer program that services clients or a computer running multiple such services. A server is usually dedicated to a task such as a web server, a print server or a database server. Any computer can be classified as a server if a server software is running in it for example a laptop which is not created for server purposes can act as a server if proper software is installed onto it. There are computers that are specifically made for server purposes and these computers have hardware that is designed for server use. This might include very fast network connections and high I/O throughput and the lack of audio and USB ports. The parts of a very important server such as enterprise servers are built with specialized hardware which has very low failure rates to optimize uptime since the cost of the server might be lower than the cost when the service is down for five minutes.(Wikipedia 2011.).

There are specialized operating systems for servers. These servers tend to have features unique for servers such as the lack of a graphical user interface, a possibility to configure and update software and hardware without rebooting or interrupting the service, excellent back up services, transparent data transfer between hard drive volumes or hard drives, excellent networking capabilities and high levels of security. There are various different operating systems for servers such as Windows Server software family and many distributions of Linux e.g. Suse Linux Enterprise Server (SLES).(Wikipedia 2011).

4.2 Client

A client is an application or a system that connects to a server to access a remote service. A good example of a client is a web browser. A web browser connects to a web server and retrieves information from that server and then displays it, the same thing is with an email client. When playing a multiplayer game the game itself is usually the client which connects to a server where other players can be seen and games created. Not always a specific client software is required as a client can be integrated into a website and thus make a web browser a universal client.(

Wikipedia 2011.).

Clients are generally classified into three different categories. The categories are fat clients, hybrid clients and thin clients. Fat clients are the heaviest as the name suggests and have local storage and local processing. Hybrid clients have local processing but no storage and a thin client is just a client with all the storage and processing done on the server side.(Wikipedia 2011.).

4.3 NAT - Network Address Translation

Network address translation is used to modifying network address information in datagram packet headers in a router device. This is done to remap IP address space to another. The primary benefit of a NAT is to allow Internet access to many computers on a private network using a single public IP address. This is especially useful since IP4 address are very limited nowadays.(Wikipedia 2011.)

There are drawbacks in sorting outgoing traffic through a translated address, as computers behind a NAT cannot have end-to-end connectivity and cannot use certain Internet protocols. Such services include active FTP, Session Initiation Protocol and Voice over IP. (Wikipedia 2011.)

4.4 Chat Room

The name chat room can be used for a range of different services including real-time online chat, instant messaging, online forums and graphical social environments. It can be a standalone software such as IRC client (Internet Relay Chat client) or a service which requires no special software such as a website or part of a website. Generally in a chat room there is a list of users currently online, a typing box where the user can type the message to be sent and a larger area where sent messages are visible. (Wikipedia 2011.)

A chat room can be a visual environment such as Habbo, earlier known has Habbo Hotel, in which users have avatars and can move around. Chat rooms can also include games such as tic-tac-toe or scrabble. In some cases a chat room can work as a game lobby where users go to find company for online games. A good example of such a service is Blizzard Entertainment's Battle.net or the now defunct MSN Gaming Zone. In these services users connect to a lobby where other players are listed, sent messages are visible and they create or join games.(Wikipedia 2011.).

5 MY PROJECT

5.1 Idea

The aim of this project was to learn the basics of Unity networking. The purpose was to have fun while learning and so the idea of creating something useful was planned, and thus the idea of creating a chat lobby where users can talk and invite other users to games was created. Essentially this is just another game lobby, but the requirement was to create it so that it could be carried on a USB flash drive and the same client could join and create servers. This way there is no need for a dedicated server as users can create a server when they need one and invite their friends over.

As this was a project to learn basic Unity networking, the intention was not to develop this to a ready-for-shipping product. Due to this some features had to be dropped such as making an example game, custom graphics for the GUI and master server support. Adding these into the project would have at least tripled the development time which was not desired.

5.2 Planning

The first thing after coming up with the idea was to draw the GUI. As, when developing something I personally want to create the base onto which I build the features. This way it is possible to keep track of what needs to be done and what needs to go where without looking at the concept papers, and it is invigorating to see the project proceed. When the GUI was drawn, step by step work was started to get the planned features implemented. These features included: creating a server, joining a server, listing users, sending messages, displaying messages and game invites.

5.3 Obstacles

Since this was a learning project the first thing to do was to read the Unity forums for tips and tricks and look for tutorials about Unity networking. A very good Unity networking tutorial created by M2H Game Studio was processed first. From this tutorial it was possible to gather plenty of information on Unity networking. After every tutorial step time was spent working on the script to see how things really work and to memorize the commands and functions. This information gathering phase took around one and a half weeks.

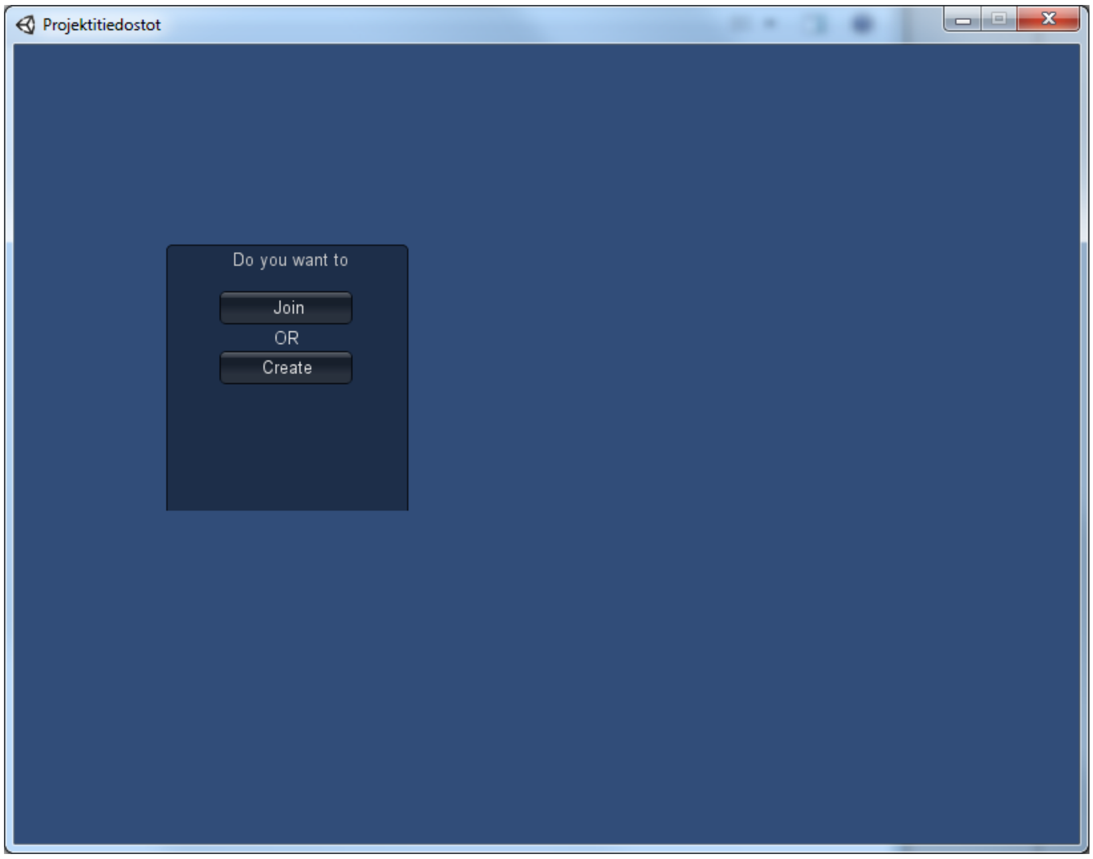
5.4 Implementation

The first step in the project was to ensure connectivity. This included several lines of code and two buttons. One was for joining a server and the other was for creating one. After the connection had been established between the clients developing the GUI for the chat room itself started. The chat room has three main components: a message display box, a message type box and listed users. The message display box is as the name suggests, an area where the user messages are displayed. The message type box is where the users type their message to be sent for others to see. Finally, the listed users area lists the connected users in the order they have connected. In addition to this the listed users area includes the game invite buttons which can be used to invite another player to a game.

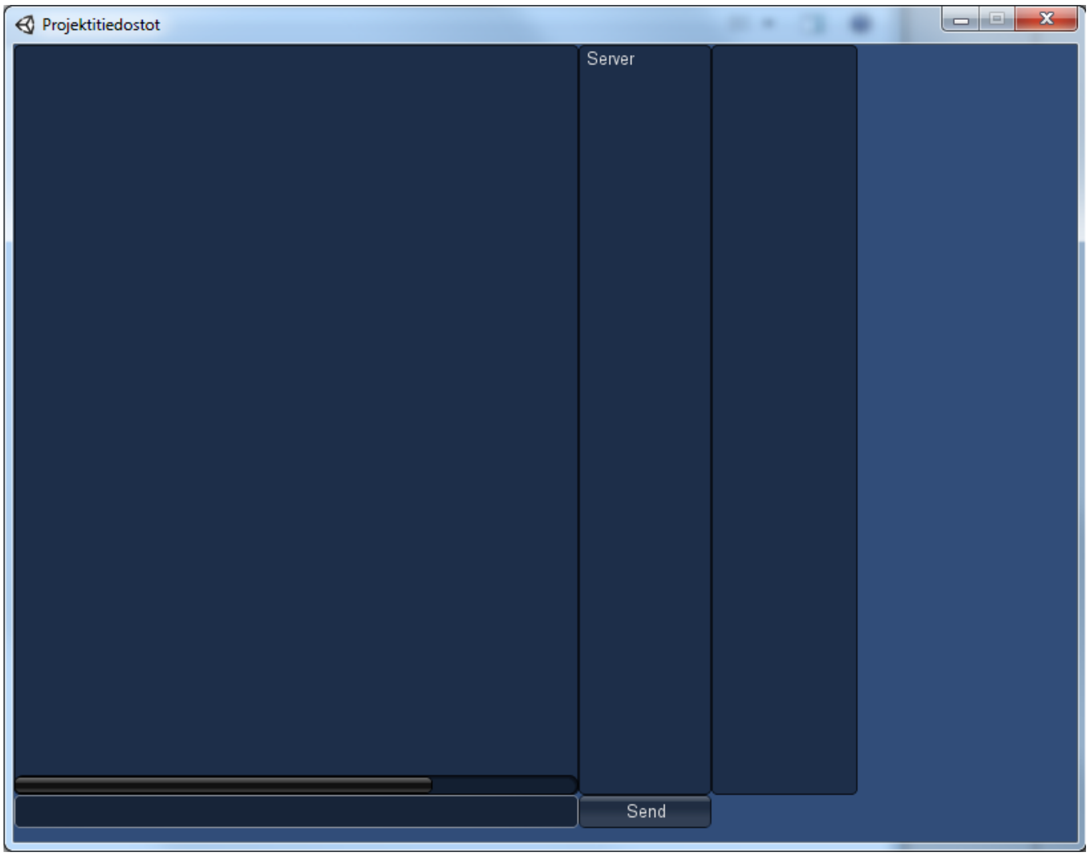
To create a server the user only has to click the Create-button in the startup menu and the server is ready to go. To join, the user has to enter an IP address, port and his/her desired name.

The chat room itself works as any other. A user types a message he or she wants to send and hits the enter-key or send-button on the GUI. There is a spam prevention check which prevents a user from flooding the message screen. Users are listed in the order they connect to the server and they have game invite buttons on their right side in the listed users display. The message display area shows if a user leaves or enters, and displays the messages the users have sent.

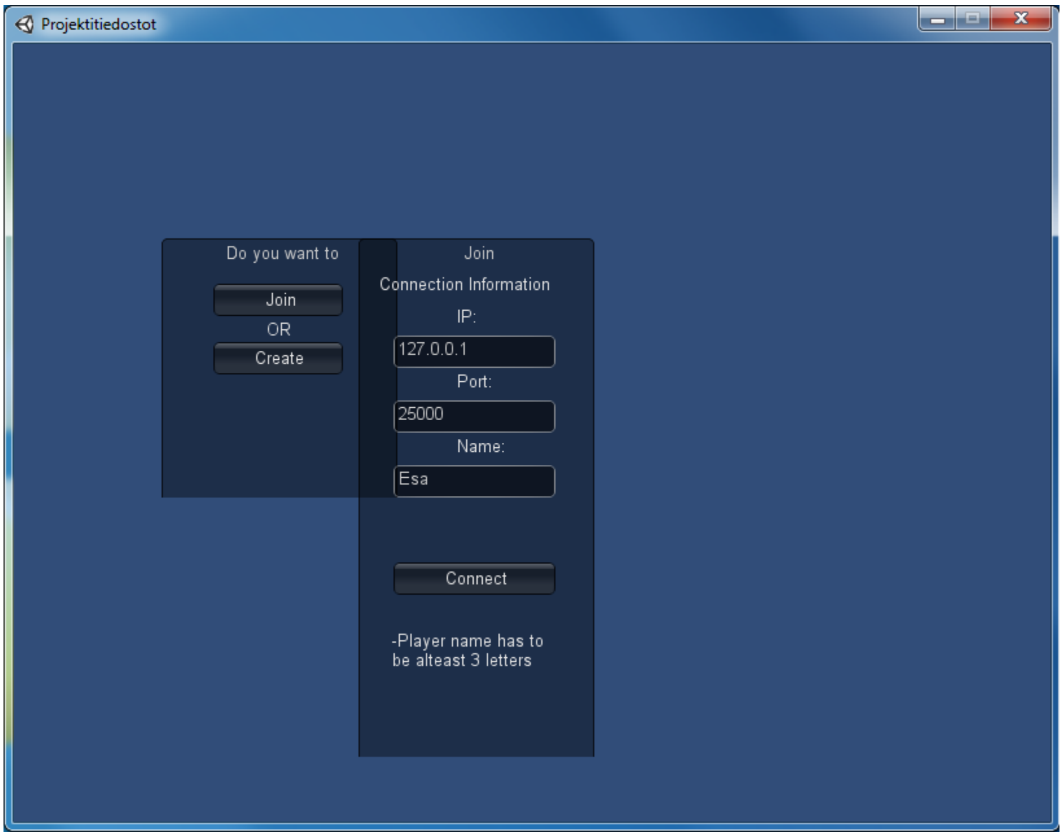
If a user wants to invite another person to a game, he or she can click the invite-button next to the other one’s name. This opens a dialog where one can choose a game to which the other is invited. Clicking OK in this dialog will send a game request for the invited user. The user who invited will be shown an information box which informs that their invitation is being processed by the invited user. If the invited user declines a game session, the invitation sender will receive a message about the declined invitation. Should the invited user accept the game invitation, both users will load the game automatically.



PICTURE 10. After the start up the user has to select whether to create or join a server.



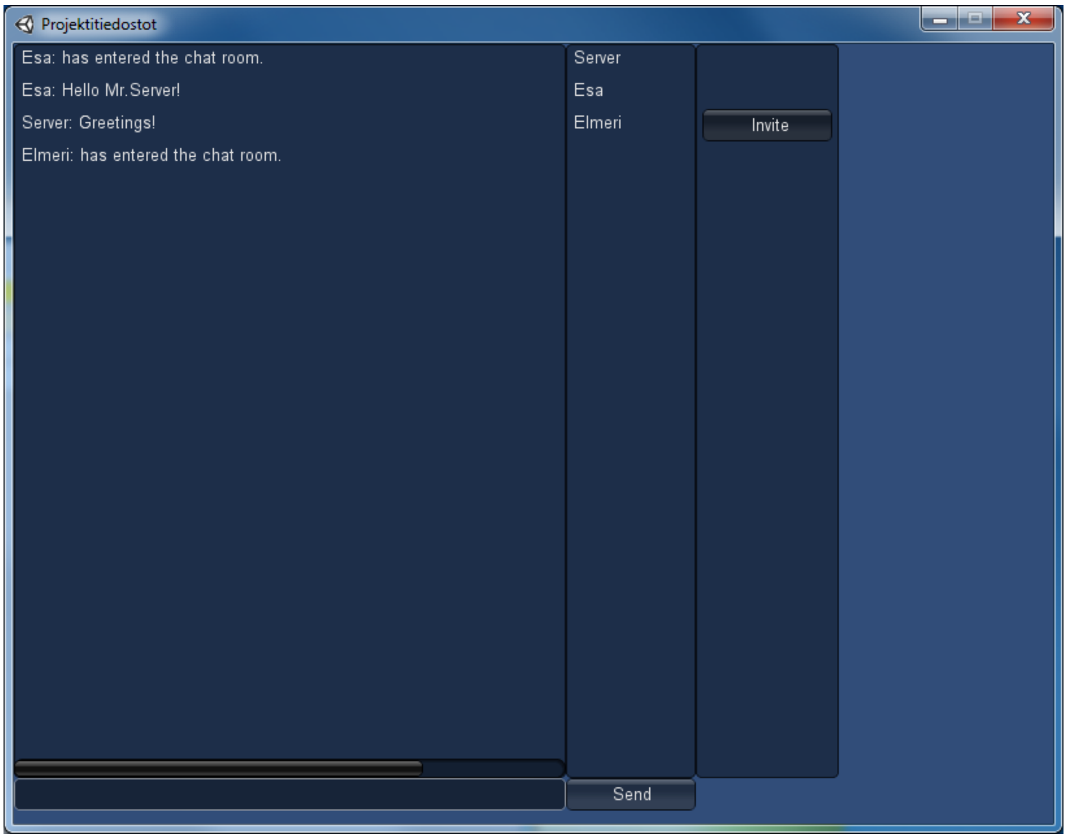
PICTURE 11. Selecting the server creates the server and it is ready to use.



IP:
127.0.0.1
Port:
25000
Name:
Esa

-Player name has to be alteast 3 letters

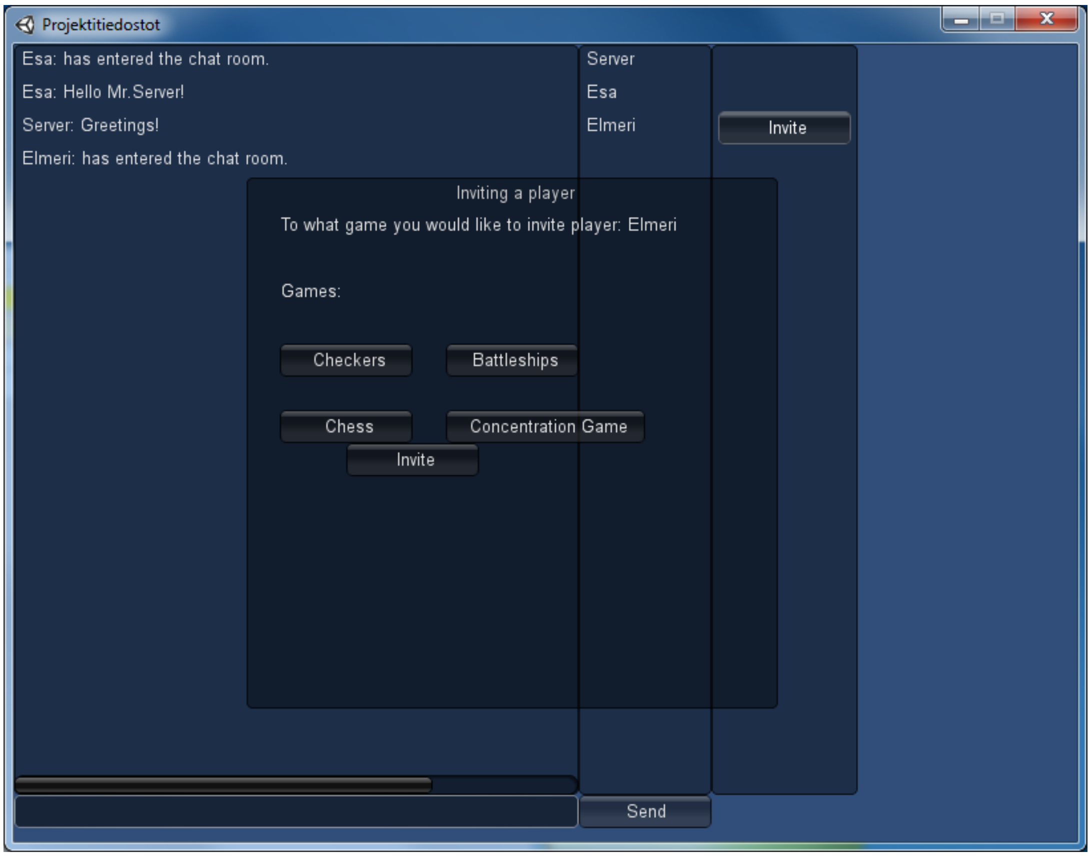
PICTURE 12. User joining a server. An IP address, port and name is required to join in.



Esa: has entered the chat room.
Esa: Hello Mr.Server!
Server: Greetings!
Elmeri: has entered the chat room.

Server
Esa
Elmeri

PICTURE 13. Two users have joined in and started to chat.



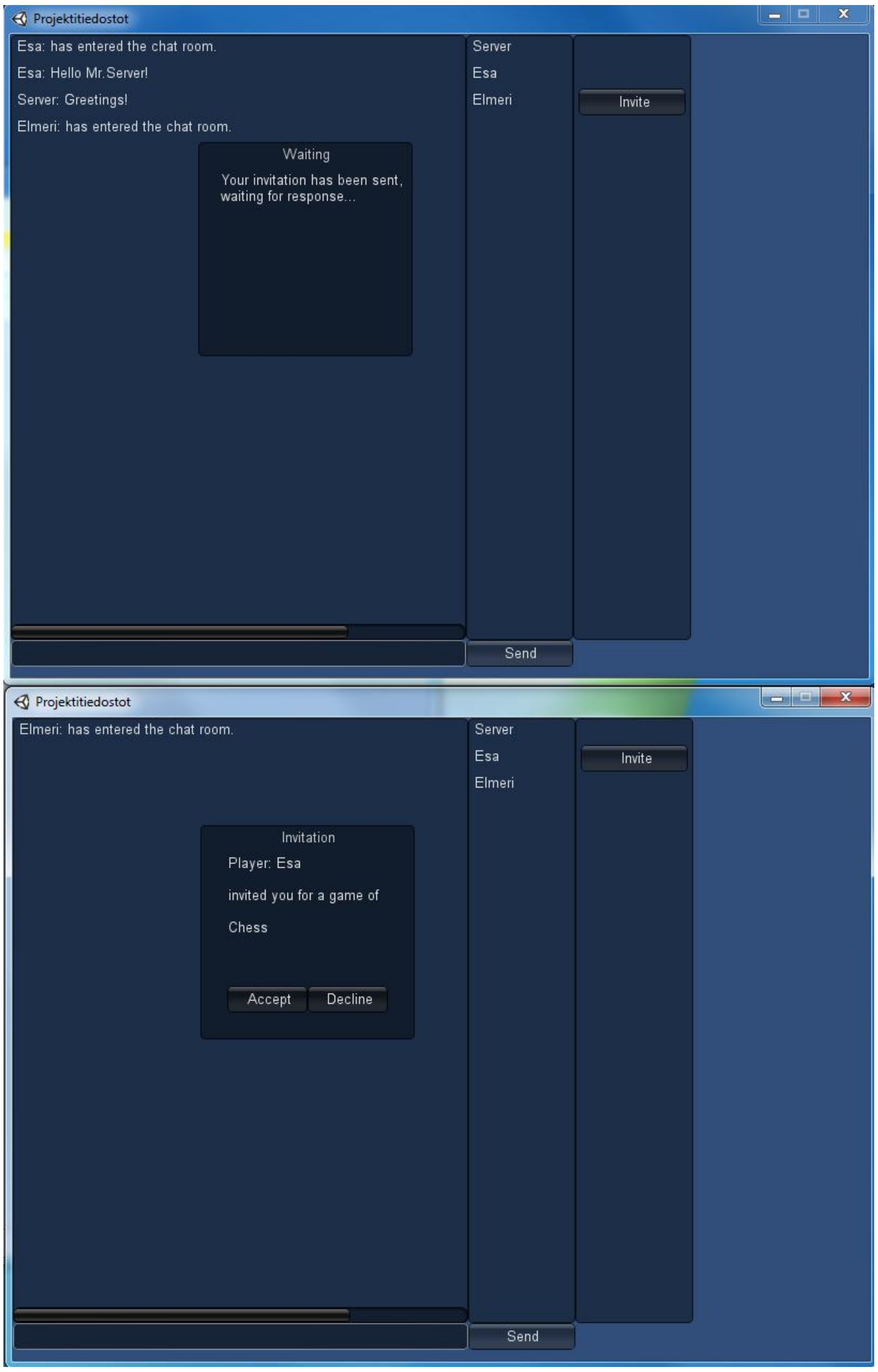
Esa: has entered the chat room.
Esa: Hello Mr.Server!
Server: Greetings!
Elmeri: has entered the chat room.

Server
Esa
Elmeri

Inviting a player
To what game you would like to invite player: Elmeri

Games:

PICTURE 14. User Esa is inviting Elmeri to a game.


Esa: has entered the chat room.

Esa: Hello Mr.Server!

Server: Greetings!

Elmeri: has entered the chat room.

Waiting

Your invitation has been sent, waiting for response...

Elmeri: has entered the chat room.

Invitation

Player: Esa

invited you for a game of

Chess

PICTURE 15. After sending the invitation, Elmeri receives a dialog where he could accept or decline it. The sender receives a response window to see that his invitation has been successfully sent.

5.5 Further Development

As mentioned, this version lacks several features which would add value for the project and the project has not been thoroughly tested. Master server support would allow dedicated servers where users could connect behind NATs and public servers could be made. Now the joining process requires an IP address and the port of the server which is not very good for mass access.

To make the application look more desirable it would require custom made user interface graphics as the default boxes and buttons are rather bland. Creating custom graphics is not something I myself can create, at least to the point where they would be any improvement from the default graphics.

Creating a multiplayer game for the chat room as an example of what needs to be sent to the game from the server to game level is something that needs to be done if this is to be developed any further. A simple game such as tic-tac-toe or a card game would suit this purpose well and would take about a month to develop. This would be the logical first step for further development since the chat room itself is not a service that people would use over the other dozens and dozens of chat rooms. Thus creating a simple game could attract users to the product.

If the project is developed further, the first step would be to create a game for the chat room. First it should be decided whether to include the game inside the client or to stream the game for the users after they decide which game to play. Both have their pros and cons. Having the games inside the client would reduce load times as no streaming is required and when no streaming is done the server computer will not be a bottleneck. Having games included in the client would however drastically increase the client size. Including the games in the client would require optimizing the games to take as little space as possible. If the games were to be streamed from the server to the clients, the server would have to have a very fast Internet connection or otherwise joining in the game would be painfully slow. The advantages of this are that only the server has to have the updated versions of the games played and the client size would stay small.

Finally, the user interfaces would need redesign to have a proper pattern and good usability, and proper testing would be required to grind out possible bugs. Now they are just created to work and very little time has been spent on considering how good they look or how easy they are to use. All in all, a lot is needed to make this a viable product but in essence the core functions are ready and the main goal, which was to learn Unity networking, was indeed accomplished.

5.6 Chat room code examples

Creating a server in Unity is very simple and can be achieved by a single line of code:

```
            Network.InitializeServer(64,25000);
```

This creates a server which listens to port 25000 and allows 64 simultaneous connections. Connecting to a server is not much harder:

```
            Network.Connect("127.0.0.1", 25000);
```

The above function connects to a server at 127.0.0.1 in port 25000. Establishing connection on Unity is as simple as that.

When a chat room is created, an admin is also created:

```
function OnServerInitialized() {

            //--- Create "Admin" chatter
            playerName = "Server";
            var newPlayer = new ServerPlayersConnected();
            newPlayer.networkPlayer = Network.player;
            newPlayer.playerName = "Server";
            serverPlayers.Add(newPlayer);
            networkView.RPC("ShowPlayers", RPCMode.OthersBuffered, "Server",
Network.player);

}
```

In the above code the player is given a default name and his or hers connection information is stored in a ServerPlayersConnected object and this object is then added to an ArrayList which keeps track of the players who are connected to the server. The last line of code buffers the creation of this server player, so that when other players join the server they are automatically informed.

Listing the connected users is done by going through all the ServerPlayerConnected objects in an ArrayList. This same method is used in client side listing and also when listing the game invite buttons. The example below is server side player listing:

```
if(Network.isServer)
 {
 for(var entry : ServerPlayersConnected in serverPlayers)
  {
  if(counterServer < serverPlayers.Count)
   {
   GUI.Label( Rect(5,25*counterServer,100,25), entry.playerName);
   counterServer++;
   }
  }
  userCounter = counterServer;
  counterServer = 0;
  }
```

Large parts of the code revolved around creating ArrayLists of objects which were handled in for-loops. These include player and message storing, game invites and creating UI elements. These lists made it easy to handle those procedures and kept them neatly organized.

6 Sources

Battle.net
http://en.wikipedia.org/wiki/Battle.net. Read on Janaury 2011.

Chat room
http://en.wikipedia.org/wiki/Chat_room. Read on January 2011.

Client (computing)
http://en.wikipedia.org/wiki/Client_%28computing%29. Read on January 2011.

Habbo
http://en.wikipedia.org/wiki/Habbo. Read on January 2011.

MSN Games
http://en.wikipedia.org/wiki/MSN_Games. Read on January 2011.

Network address translation
http://en.wikipedia.org/wiki/NAT. Read on January 2011.

SUSE Linux Enterprise Server
http://en.wikipedia.org/wiki/SUSE_Linux_Enterprise_Server. Read on January 2011.

Server (computing)
http://en.wikipedia.org/wiki/Server_%28computing%29. Read on January 2011.

Seyler,B. 2010. Marketwire: Unity Technologies Unveils 'Union'
http://www.marketwire.com/press-release/Unity-Technologies-Unveils-Union-New-Division-Headed-by-Brett-Seyler-1350973.htm. Read on January 2011.

UNITY
http://unity3d.com/unity/editor/. Read on January 2011.

UNITY: Android
http://unity3d.com/unity/publishing/android. Read on January 2011.

UNITY: Asset Pipeline
http://unity3d.com/unity/editor/importing. Read on January 2011.

UNITY: Asset Server
http://unity3d.com/unity/editor/asset-server. Read on January 2011.

UNITY: Asset Store
http://unity3d.com/unity/editor/asset-store. Read on January 2011.

UNITY: Audio
http://unity3d.com/unity/engine/audio. Read on January 2011.

UNITY: Built-in Shader Guide
http://unity3d.com/support/documentation/Components/Built-in%20Shader%20Guide.html. Read on January 2011.

UNITY: Deferred Lightning Rendering
http://unity3d.com/support/documentation/Components/RenderTech-DeferredLighting.html. Read on January 2011.

UNITY: Extending the Editor
http://unity3d.com/support/documentation/Components/gui-ExtendingEditor.html. Read on January 2011.

UNITY: Fast Facts
http://unity3d.com/company/fast-facts. Read on January 2011.

UNITY: Image Effect Scripts
http://unity3d.com/support/documentation/Components/comp-ImageEffects.html. Read on January 2011.

UNITY: Lightmapping
http://unity3d.com/support/documentation/Components/class-LightMapping.html. Read on January 2011.

UNITY: Networking
http://unity3d.com/unity/engine/networking. Read on January 2011.

UNITY: Occlusion Culling
http://unity3d.com/support/documentation/Manual/Occlusion%20Culling.html. Read on January 2011.

UNITY: Physics
http://unity3d.com/unity/engine/physics. Read on January 2011.

UNITY: Programming
http://unity3d.com/unity/engine/programming. Read on January 2011.

UNITY: Ragdoll Wizard
http://unity3d.com/support/documentation/Components/wizard-RagdollWizard.html. Read on January 2011.

UNITY: Rendering
http://unity3d.com/unity/engine/rendering. Read on January 2011.

UNITY: Scene Construction
http://unity3d.com/unity/editor/scenes. Read on January 2011.

UNITY: Screen Space Ambient Occlusion
http://unity3d.com/support/documentation/Components/script-SSAOEffect.html. Read on January 2011.

UNITY: Shadows in Unity
http://unity3d.com/support/documentation/Manual/Shadows.html. Read on January 2011.

UNITY: Shop
https://store.unity3d.com/shop/. Read on January 2011.

UNITY: Terrains
http://unity3d.com/unity/engine/terrains. Read on January 2011.

UNITY: Tree Creator Guide

http://unity3d.com/support/documentation/Components/class-Tree.html. Read on January 2011.

UNITY: Wheel Collider
http://unity3d.com/support/documentation/Components/class-WheelCollider.html. Read on January 2011.

UNITY: Writing Surface Shaders
http://unity3d.com/support/documentation/Components/SL-SurfaceShaders.html. Read on January 2011.

Unity - Building the Master Server/Facilitator on your own
http://unity3d.com/support/documentation/Components/net-MasterServerBuild.html. Read on January 2011.

Unity - Master Server
http://unity3d.com/support/documentation/Components/net-MasterServer.html. Read on January 2011.

Unity - Network View
http://unity3d.com/support/documentation/Components/class-NetworkView.html. Read on January 2011.

Unity - RPC Details
http://unity3d.com/support/documentation/Components/net-RPCDetails.html. Read on January 2011.

Unity - Unity Web Player and browser communication
http://unity3d.com/support/documentation/Manual/Unity%20Web%20Player%20and%20browser%20communication.html. Read on January 2011.

Unity Script Reference - Gizmos
http://unity3d.com/support/documentation/ScriptReference/Gizmos.html. Read on January 2011.

Unity Script Reference - MonoBehaviour.OnSerializeNetworkView
http://unity3d.com/support/documentation/ScriptReference/MonoBehaviour.OnSerializeNetworkView.html. Read on January 2011.

Unity Script Reference - WWW
http://unity3d.com/support/documentation/ScriptReference/WWW.html. Read on January 2011.

7 Appendixes