Introduction

In this document I use my latest project, Cubelands (http://www.Cubelands.com), to discuss user customizable virtual worlds. Cubelands is an online multiplayer building game I've started creating in the summer of 2010. User created content is a hot item todays games, in Cubelands this also is the most important aspect of the game. This document describes the development of Cubelands and focuses mainly on the social/community features of the game.

This paper has been written for *Creative Technology, Universiteit Twente* and *Multimedia, VU Amsterdam*.

Mike Hergaarden
http://www.M2H.nl
September 2010

Table of contents

The idea

Before discussing any community or technical aspects of games in general it is important that you're introduced to Cubelands as I use the experiences of this game in the entire document. You'll quickly understand what Cubelands is all about as it can be summarized in five main items.

1. Social gaming + CO-OP (Cooperative gameplay)

I have always loved multiplayer co-op games, I find these type of games way more enjoyable with friends than ordinary multiplayer games. This preference heavily influences the games I've made in the past years; Nearly all featured multiplayer, but all of them had at least some community integration.

In Cubelands I wanted to take the social gaming aspect a step further compared to my previous games. Besides multiplayer cooperative gameplay I've now also added a tight integration of the games community;

- Players can register a free Cubelands account in-game. They are automatically logged in to the website
- The forum is seamlessly integrated to the website and requires no registration or login, you login just once to the website and the game.
- Players have player profiles on the web and can write messages on each others profiles similar to Hyves and Facebook profile pages.
- Players can upload screenshots to their online profile from inside the game, to show off. You can of course comment on these screenshots. Furthermore the screenshot page encourages players to share their screenshot on external websites(Twitter, Facebook, etc.)
- There is a friends list both in-game as on the website. Players can use this to see who's online and quickly join a friends game.
- Achievements... it's all about the bragging rights.
- Game servers are run by the community. Users are given tools to manage Cubelands servers.

The social aspect of the game will be expanded in the future. One of the first additions that's planned are community contests. The community contests will be all about building the best Castle, boat, car and so forth. One great feature of these contests is that these contests can be automated: A script will add a random contest from the list every once in a while, and decide on the winners based on community voting. I'll get back at making good use of the community in the *Community* chapter.

2. User made content

Making games is a lot of work, mainly content wise. So instead of creating a lot of content it can be wise to invest time in tools to enable the community make their own content. This allows for unlimited hours of gameplay. In Cubelands the tool to create content is the game itself. Players can remove/add cubes in the entire world and share their creations via screenshots. These screenshots are automatically uploaded the the games community website.

You do need to be able to 'protect' your own creations otherwise the hard efforts are not worth it. The 'value' of creating something would be less if anyone could easily copy it or destroy it without nearly as much effort. Cubelands allows for creating a world using predefined blocks. However, if players run their own servers they can customize the following settings:

- Moderation: Banning/Kicking players
- Ability to custimize the cube textures (and therefore the look of the entire world)
- Change world size
- Maintain a list of who is allowed to build on this server
- Private/public server (passwords)

3. Author once, deploy everywhere!

To quickly reach a wide audience I've planned to deploy on a huge range of platforms. Thanks to the Unity engine this is easily possible. At the moment the game is available online on Cubelands.com and Facebook plus you can download a standalone player for Windows and Mac.

Within a few months we I want to have deployed to the following platforms:

- Webplayer (Mac + Windows)
  - Cubelands.com
  - Facebook app
  - MySpace app
  - game portals: Wooglie.com, GameJolt, ...more?
- Standalone game (Mac + Windows)
- Mac Dashboard widget
- iPad

Unity allows me to seamlessly create one project and deploy to all these platforms in just one click. The only platforms I might want to customize the game for are Facebook and iPad. Facebook could require some customization to make better usage of the community sites features (invite friends, etc.). The iPad needs some

extra custimization though; I have yet to see if the iPad can run the game properly. Furthermore the controls need to be customized for the iPad. Platform specific customization can be easily done by using the following code:

```
//See http://unity3d.com/support/documentation/Manual/Platform%20Dependent%20Compilation.html
void MyPlayerInputFunction(){
  #if UNITY_IPHONE
    Debug.Log("iPad movement code here");
  #endif
  #if !UNITY_IPHONE
    Debug.Log("normal input code here!");
  #endif
}
```

4. Commercial aspect

Selling the game
All of my previous games have had a free version. This is a great free marketing tool. As online advertisements cover the costs of all the free players there's no real downside. However, Cubelands will be my first 'real' commercial game (ignoring some simple iPhone apps that never really lifted off). The first few months of developments the full game can be bought for 7.50€ once the beta finishes the game will cost 15€. I've avoided a membership option on purpose as I believe it'll be much easier to sell a single game license then to convince someone to buy a membership (you never really own the game).

In the future I might add microtransactions to allow players to buy customizations for their avatar (say a fancy hat for 0.50€).

Payment provider
While I've previously used just PayPal for selling games, this time I'm using a 3rd party payment provider. Such a provider takes a bigger share than PayPal, but it's worth it. Outsourcing the payment portal saves you the hassle of handling fraudlent payments, refunds and finally this third party allows you to support multiple currencies, languages and payment options.

5. Freedom

Games are becoming more and more immersive. You are sucked in to a games story; mostly by allowing you to to customize your experience by making a lot of choices that affect the game. Having all these choices can also really limit you; i.e. having chosen to play as a mage in a RPG game disables a lot of options in

the game (using a sword, etc.). I dislike that you can regret the choices you've made. For Cubelands I want true freedom. You don't choose a specialisation or whatsoever.

There's just one problem I still need to address regarding all this freedom: For some players the freedom of Cubelands is confusing. Players expect the game to tell them what to do. Therefore I do plan to add some traditional game modes inside the world of Cubelands.

Community

In past projects I've learned a lot about community management. Therefore I hope to apply these lessons to Cubelands, this game is all about the community after all.

Community management is a big task

Community management is hard. People hate or love your game and this can vary by the hour. Your biggest friends can easily become the biggest troublemakers. There's two options; either you don't get too much involved in the community OR be very open and clear. Be aware that the users will keep you to your words, so set clear boundaries whenever you promise something. If there are any problems you might try to hide these from your users and try to solve them (with a good intention) . But in the end you'll cut yourself in the fingers as they will always find out. Your players will prefer you to play open cards, you'll be amazed to find out what they can cope with! When designing a game it's important to design a plan for the community (management) as well.

Self managing community

Managing a game is a big task; Starting up a new game is very easy, updates can be applied in seconds. However, how bigger the game gets (player wise) you risk a big slowdown regarding development. This is mainly because of community management. Moderating the users content is a day job. Therefore you should design a game so that the community can moderate their own content (Be it messages, screenshots, etc.).

In Cubelands the user generated content are screenshots, screenshot comments, profile comments and in-game constructions. For the comments I enable the author and the profile owner to delete comments. So players can moderate their own pages. Furthermore I allow the community to flag screenshots and comments as inappropriate. Moderators that I've picked from the community can then easily see a list of screenshots and comments that are under review for deletion. Furthermore I've given server admins tools to assign moderators (and more admins) for their servers so that they can moderate their servers themselves. Even though there have been some invasions already (i.e. a lot of swastika screenshots), they were dealt with quite quickly.

The bottom line is to empower users: Reduce moderation work for yourself and make the users feel like they have a real influence on the game.

Cubelands community portal

Ask your users for feedback

You should use your users to shape the future of the game; they roughly know what they want(not always) so you can ask for their feedback. Again, this feedback should sort itself out. There are user feedback sites that allow users to psot feedback and vote on importancy. These sites automatically detect ideas that have been reported before, allow merging, update feedback status, (user) moderation etc. Cubelands uses uservoice, see: http://cubelands.uservoice.com/.

Do the social media hypes work?

The social media sites that Cubelands uses are twitter, facebook, uservoice, and blogspot. I have put serious thought into the combination of these sites to ensure I'm not wasting time. Each of these sites has a real purpose and is used accordingly. I sue the blog for all development news and other 'articles'. The blog is linked to facebook and twitter so that I don't have to update twitter with new blogposts and I don't have to update facebook at all. Twitter is used for random, short information. Twitter and Blogpost are actually a joy to use. They keep your users close and keep your own motivation high. I don't update Facebook but the Facebook page does list blogposts and can be used by the users to 'like' Cubelands. Lastly I have already explained the use of uservoice in the previous section. Concluded; you really want to ask your users for feedback and you need to publish your news somehow; so maintaining a blog, twitter, uservoice is a good idea.

Technical realisation

When playing Cubelands you might not realize it right away, but there are up to one million cubes on your screen at the time and a whole world contains millions of cubes(server side) Getting those rendered as efficient as possible was one of the most important technical hurdles of the game.

A player made Pyramid made up of ten thousands of cubes

Performance limits

Cubelands is affected by the following performance limits:

Network bandwidth (Server)
: The server has a network traffic bottleneck since it routes all networking traffic between all clients (no P2P).

Rendering/GPU (Clients)
: A client can only display as much drawcalls, triangle/vertices as the videocard can process

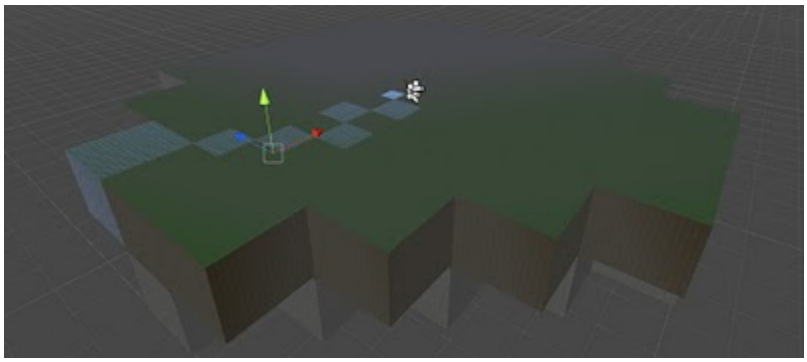
Memory (Server)
: On servers there's a problem with the worldsize not always fitting in the main memory

CPU: Saving/loading (Server + Clients)
: Saving/loading worlds costs quite some time on both client and server when a new client joins.

The solutions

World streaming
Affects client & server network traffic, improved client memory usage.

The world is streamed to clients. The client 'register' and worldchunks that they are interested in. The server will send them the entire chunkdata, and will notify them of future updates. At the moment the chunk size is 32x32x32 cubes. Client register a chunk when they are closer than chunksize*5 and they unregister a chunk when they are further than chunksize*7. This is to prevent rapid registering/unregistering as registering a new chunk is quite heavy for both client and server.



Only the chunks close to a player are registered

Rendering millions of cubes:
Affects client GPU usage

There are two main bottlenecks for rendering in Unity games; the amount of draw calls and the amount of triangle. I'm ignoring vertices here as it *usually* goes hand in hand with the triangles count.

Cubelands reduces drawcalls by simply using one and the same material for the entire world. Because of this entire world could be rendered in one drawcall, however, to enable culling chunks are sized 4x4, 8x8, 16x16 or 32x32. The vertices count is kept as low as possible by only drawing the faces of cubes that are possibly visible (only if there's no neighbour cube at that side). I have not implemented a way to detect wheter these sides are impossible to see because, for instance, they are inside a cave. I guess these calculations are not worth their costs as, on top of the calculation costs, this would also require the game to regenerate certain cubes dependant of your position.

The drawcalls are no problem per se, however, users are able to change cubes which makes things a bit more complicated. For rendering it would be best to have huge meshes (less drawcalls). But if a user changes a single cube the entire mesh has to be rebuilt. Therefore the chunk size is in relation to how far they are from a player (and thus how likely they are to change). This dynamic mesh size is displayed in the image from the previous item.

Dump pagefiles - not yet implemented
Affects server memory (crashes!)

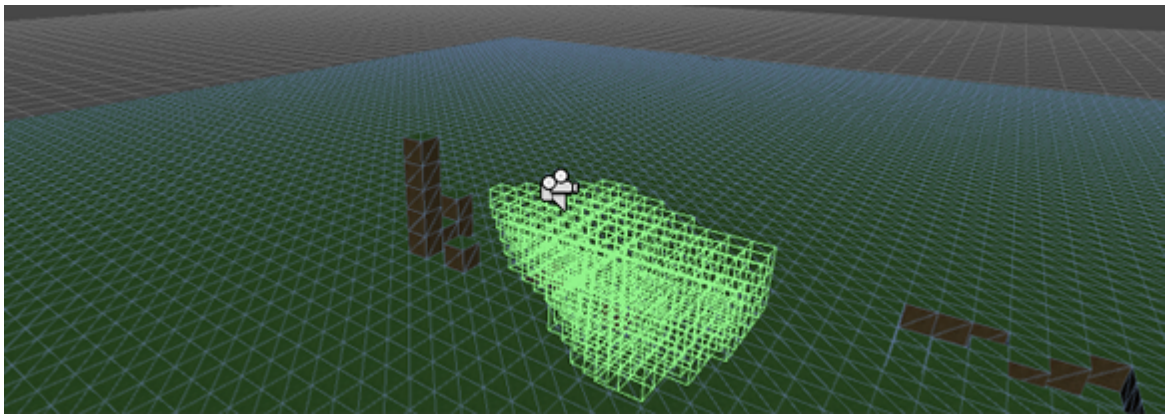
This is the only performance issue that has not yet been addressed at the moment of writing. Thanks to the world streaming the clients could now explore an infinite world, as long as the server supports it. However, servers crash when using a big world size because the world doesn't fit in memory. A solution to this problem would be to have the server only load the worldchunks in memory that the players are using. There still needs to be a safetycheck here because players could spread out and crash the server.

Saving & loading
Affects client&server CPU performance

Saving and loading the world can be quite heavy and can easily take 3 full seconds in which a server cannot process other requests. Quick world loading has been solved by the streaming world feature, the client load time was reduced from 20+ seconds to just a second. Server side saving is still taking quite some time but is done in the background so that the game does not suddenly lag. The save could be a tiny bit inconsistent with the world, but that's no issue in this case.

Physics

Affects client CPU performance



The physics move with the player

In Cubelands the physics are calculated client-side only to save bandwidth and performance on the server. Still, the Unity physics implementation required a hack to function well. It's too heavy to simply add physics to all cubes. Physics are used for movement and for detecting the cube that you're aiming at.

At the moment physics are added for all cubes that are 6 cubes distance from the player. There's room for improvement here by faking the physics and doing the calculations manually. I believe the Unity implementation is overkill in this situation. Spawning and removing the physics for all cubes on the fly might be heavier than necessary.

Other technical topics

Multiplayer

For the multiplayer I chose the setup I am most comfortable with, plus it's the cheapeast solution. I'm using the built in Unity networking. This is a cheap solution since I only have to host a masterserver that keeps track of all player hosted games. It's using a client-server model.

For more information about Unity networking see my Unity networking guide at http://www.m2h.nl/unity/.

Multi platform patching

I currently deploy to three targets: Webplayers, Window standalone and Mac standalones. Mac dashboard widgets and iPad are planned for the official launch.

The webplayers and the Mac dashboard widgets are easy to update as they are hosted on my own server. Players automatically download the latest version. The iPad version can be updated via Apples app store.

This leaves us with the standalones for both platforms. I've looked at update/patcher

I will be releasing this tool to the public, keep an eye out on my Unity resources page at http://www.m2h.nl/unity.

Generate procedural meshes

The cubes that make up the world are procedurally generated at runtime. The code below shows you how to do so. Only the lengthy verts/tris/UV calculation is left out.

```
Mesh mesh = new Mesh();
mesh.vertices = verts;
mesh.uv = newUVs;
mesh.triangles = tris;
mesh.RecalculateNormals();

GameObject go = new GameObject("MyMesh");
go.transform.parent = parent;
go.transform.localScale = Vector3.one;
go.transform.localRotation = Quaternion.identity;
go.transform.localPosition = Vector3.zero;

go.AddComponent("MeshRenderer");
go.renderer.material = myMaterial;

MeshFilter filter = go.AddComponent(typeof(MeshFilter));
filter.mesh = mesh;
```

Generating a custom mesh

One very Unity important tip: When cleaning up custom made meshes don't forget that besides calling Destroy(); on it's GameObject/Meshfilter, you need to explicitly call Destroy on the mesh that you've generated. This is by design. If you forget to do so you'll have a huge memory leak. Cubelands used to leak up to hundreds of Mb memory until a inevitable crash cleared

the memory.

```
Destroy(myGameObject.GetComponent<MeshFilter>().mesh);
Destroy(myGameObject);
```

Destroying a generated mesh

Streaming/Assetbundles
No streaming required: no data really.

In-game building

For a perfect in-game building example that shows you how to allow users to custimze a world you can check out the "Domino builder" example project at http://M2H.nl/unity.