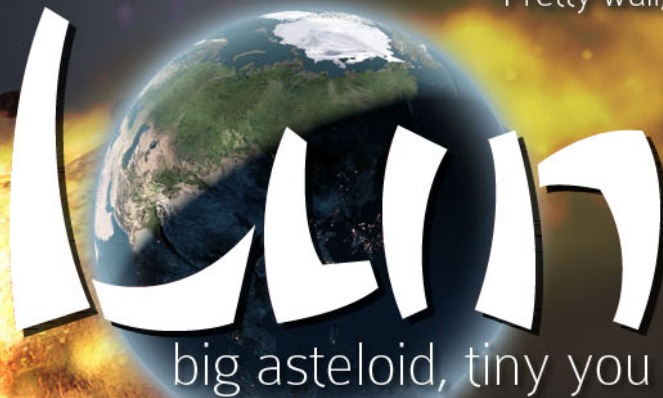


Interim Report

Historic Places and Events:
Pretty wall, pretty fast!



Lun
big asteloid, tiny you

Table of Content

Table of Content	2
Current State	3
XNA tryouts, getting started with the environment	3
Exploring possibilities for production pipeline	3
Preliminary tests for concatenation functionality	3
Graphics data Classes	3
Basic blocks, player sphere, simple doom	4
Track data structure / class	4
Initial texture mapping	4
Texture drawing	4
Physics	4
Bounding geometry	5
Terrain	5
Simple track pieces (turns, up, down, hole)	5
Coin model + texture	5
Tower block modeling + texturing	6
Environment cylinder	6
Slightly improved doom	6
Sound	6
Particle system	7
Visual Impressions	8
Startscreen	8
Track, Gap & Tower	8
Environment	9
Monster Ball	9
The approaching Doom	10
Tasklist updated	11



Current State

The time schedule presented earlier in the lecture states that we are in a rather good timing situation. We added all content from functional minimum and the minimum target is implemented and running. Right now we are deep into desired target and we are all looking forward to see the finished game.

But we had to find out that we do have some problems with the garbage collector, as we are creating too much overhead. We're now profiling the game and reducing overhead as much as possible.

XNA tryouts, getting started with the environment

Using XNA requires some sort of knowledge of the environment, so we spent some time familiarizing with its structure and creating simple tests to see how the basic stuff works, mainly importing and drawing geometry and input handling.

Exploring possibilities for production pipeline

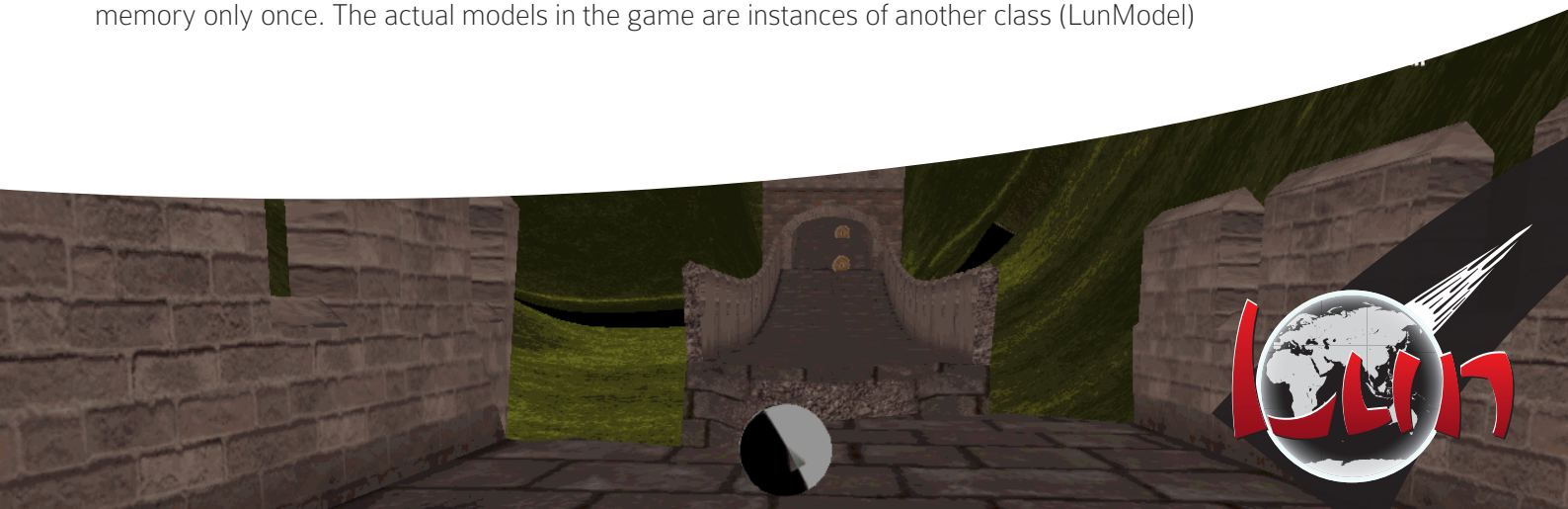
Before starting with the production of game assets it is wise to define clearly the production pipeline. With the help of the previous task we identified the .FBX as our format of choice to export geometry from 3D Studio to XNA. Various tests helped us pinpoint all the details like unit scaling and animation exporting.

Preliminary tests for concatenation functionality

A fundamental functionality of our game is the concatenation of single track pieces, so we tested it very early, and it turned out to be a good thing, since the recovery of specific points inside the imported meshes (marked by non-rendered cubes strategically disposed around the main mesh) wasn't exactly straightforward.

Graphics data Classes

The data of the imported meshes is stored with a dedicated class (LunModelData) that holds a single copy of the base data of each different mesh: this ensures that the data is loaded into memory only once. The actual models in the game are instances of another class (LunModel)



each representing a different one (each with its own transformations and positioning) and a referencing the base data. We also implemented a library class that handles the loading and the instantiation.

Basic blocks, player sphere, simple doom

With the class framework in place we produced the very basic 3D assets: a straight wall piece, the player's sphere and a simple plane representing the doom, to use during the implementation of the functional minimum.

Track data structure / class

The track of a game level needed a container class to hold it and to perform the needed operations on it once it has been created, as the repositioning of each track piece (that needs to be put exactly after the previous one).

Initial texture mapping

The basic models got an UV texture mapping and got re-exported to the game.

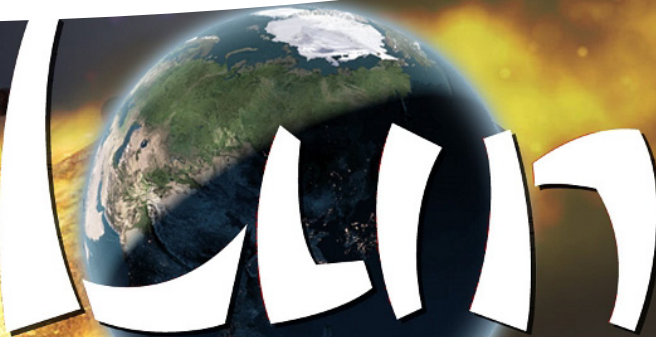
Texture drawing

The natural progression after the texture mapping was the actual texture drawing, done in Photoshop, using free textures and pictures available online.

Physics

We decided very early to use an existing physics engine, knowing that this is quite a task – especially as we're using quite complex terrain and track shapes. We're using "JigLibX" a C# port of the JigLib physics engine. It's open-source and suits our needs quite well as it allows for vertex-based collision faces as well as primitive based collision faces.

Exit
Credits



Bounding geometry

To increase the performance of the physics engine we decided to embed an invisible, simplified mesh inside each track piece to act like its “physical body” to the ends of collision detection; this simplified mesh allows us to reduce by quite a bit the number of collision checks. A secondary effect of this approach is that we can easily flag the surfaces on which the player is able to jump. And we can have different behaviour for wall and runway collisions.

Terrain

Starting from a demo (<http://www.dhpoware.com/demos/xnaTerrainNormalMapping.html>) of random generated terrain with normal mapping. We adapted the created height map to the Shape of the track. And tile the created maps along the running direction.

It works quite well so far, but needs some improvements. For one we need to make sure the tiles fit perfectly together. The connection edges already adapt to the previous tile, but there are still some misplacing due to the fact, that the tile has an multiple of an integer size, while the track ends at arbitrary points.

Next we will make sure the generated terrain blends nicely with the background terrain so the feeling of being in a rather unlimited world should improve.

Simple track pieces (turns, up, down, hole)

After the basic ones it was time to add some simple track pieces, like turns to the left, right, up and down, as well as a straight piece with a hole on it. It wasn't that hard since the straight piece was designed to be easily deformed afterwards.

Coin model + texture

While the functionality for coins and the relative modifiers was being implemented we needed to introduce also a proper model to represent them. It's quite simple, just a large, flat cylinder with a square hole in the middle.



Tower block modeling + texturing

The first of the more complex track pieces was the tower, which also marks the beginning of a new difficulty level inside the game. It was not possible to recycle existing geometry in this case, so it took a bit more time to create it.

Environment cylinder

We started generating the environment surrounding the track by adding a huge cylinder around the camera of each player that sits on the distance and shows a mountain panorama.

Slightly improved doom

The original doom representation didn't cut it anymore, so we decided to refine it: from a small flat plane to a huge flat plane :D . No, actually we also prepared a more suitable texture, but there wasn't much time left before the interim presentation so we decided to stick with that, also because the implementation of a better looking doom requires the completion of one of the next steps.

Sound

When we sat together and discussed what makes a game playable and what are the coolest features in other games which we should not miss to implement ourselves. One of the first things that came in mind, were the many different radio stations in GTA. There the background music really makes a cool gadget. We decided to try to implement something similar or at least in the same kind. Right now we do have two different music tracks, a CD full of Chinese children singing Chinese songs, which fits the location of our game and is a rather relaxed background music. For more energy driven background music we found the soundtrack of an old game called Jets'N'Guns which is performed by a band called "Machinae Supremacy".

Pity we had some performance issues at first. Which was rather quickly solved. It was mainly a bug in the "MediaPlayer". This device is not fully integrated in Windows7.



Particle system

To add an initial bit of eye-candy we implemented a simple particle system, which spawns billboards with a semi-transparent texture on it. We implemented it keeping in mind the guidelines shown in class, so there is really only one instance of the particle, referenced n times by the handler. The basic functionality was then extended with particle lifetime, rotation and scaling, to add a bit of variety. In the game the only current use is the puff of dust when the player hits a wall, but it will be used extensively in the future.

Score: 20



Visual Impressions

Startscreen



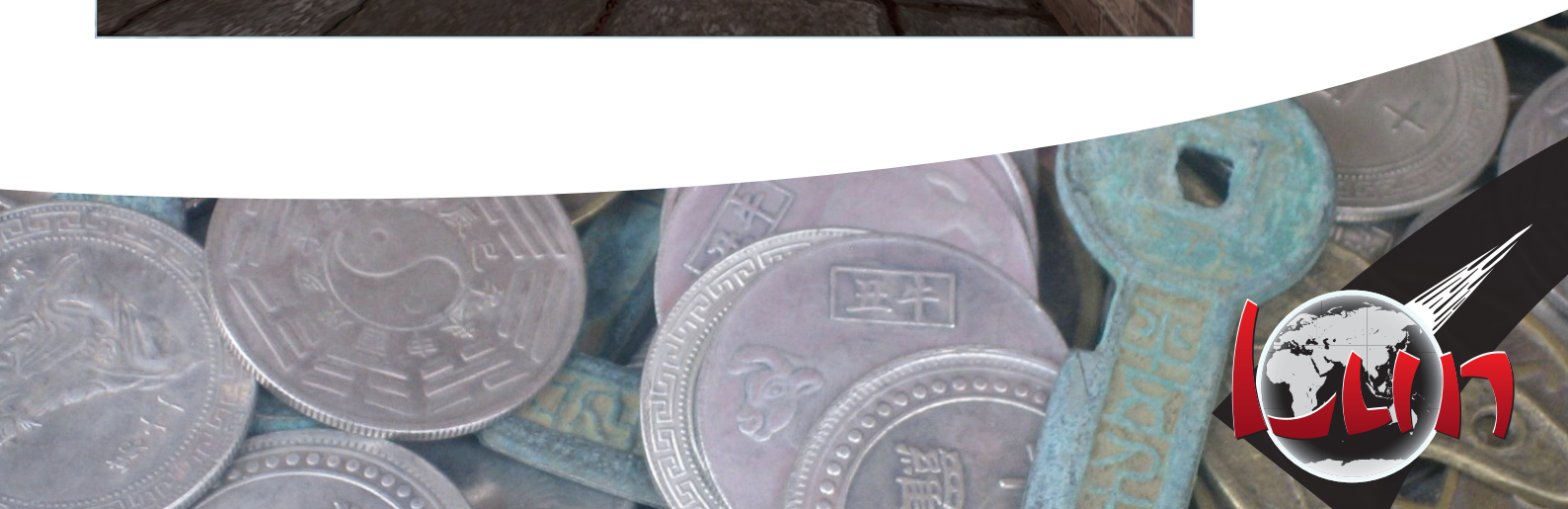
Track, Gap & Tower



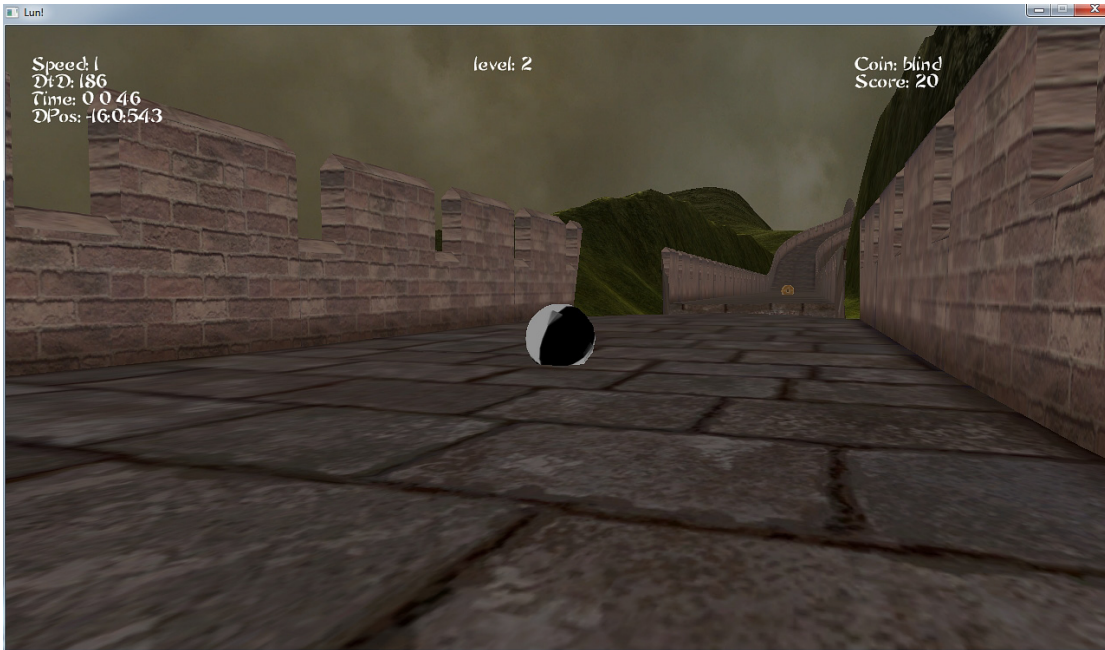
Environment



Monster Ball



The approaching Doom



Tasklist updated

	Functional Minimum	Low Target	Desirable Target	High Target	Extra
Gameplay	<ul style="list-style-type: none"> - Run over Wall - Jump over Holes - Approaching Doom 	<ul style="list-style-type: none"> - Complex Track - Multiplayer (2) - Simple Powerups - Visible Doom - "Lunscreen" 	<ul style="list-style-type: none"> - Complex Powerups - Destructible Track (predefined) - Complex Doom with Near-Death FX - Damage - Character Development (Speed, Control, Acceleration, Jump) - Intro Sequenz - Save Progress/Highscore - Multiplayer (4) 	<ul style="list-style-type: none"> - Dynamic Fractures - Visualize Damage - Global Ranklist - Post to Facebook - Different Shapes of Characters - Credits - Pause Game 	<ul style="list-style-type: none"> - 3D Mode - AI-Enemy - Different Settings/Places - Story Mode - Human charactera
Playmodes	<ul style="list-style-type: none"> - Lun! 	<ul style="list-style-type: none"> - Lun togheter 	<ul style="list-style-type: none"> - As far as possible - Ride the doom 	<ul style="list-style-type: none"> - Coin Collector 	
Graphics	<ul style="list-style-type: none"> - Basic Wall - Player (Sphere) 	<ul style="list-style-type: none"> - Simple Shaders (BM,...) - Particle FX (Smoke, Fire) 	<ul style="list-style-type: none"> - Complex Shaders - Motion-Blur - Heat-Distortion - Powerups change Player/Environment - Floating Stuff in the air (e.g. Ash Particles) 	<ul style="list-style-type: none"> - HDR-Glow - Bloom - Lensflare 	
Physics	<ul style="list-style-type: none"> - none 	<ul style="list-style-type: none"> - Friction for Player - "Sliding" - Physics Engine 	<ul style="list-style-type: none"> - Different Ground Structures - Simple Obstacles 	<ul style="list-style-type: none"> - Destructible Wall/Player - Debris Collision 	
Environment	<ul style="list-style-type: none"> - Basic Backgroundsphere 	<ul style="list-style-type: none"> - Fitting Backgroundimage - Parallaxscrolling - Generated Terrain 	<ul style="list-style-type: none"> - Changing Background (Trees-on-fire) 		
Sound	<ul style="list-style-type: none"> - none 	<ul style="list-style-type: none"> - Backgroundmusic - Simple FX 	<ul style="list-style-type: none"> - Matching Effects 	<ul style="list-style-type: none"> - Select different Styles - Progressive Sound 	<ul style="list-style-type: none"> - Surround
Powerups	<ul style="list-style-type: none"> - none 	<ul style="list-style-type: none"> Completely Rescheduled: - Blind - Monster Ball - Inverse Steering - Inverse Look 	<ul style="list-style-type: none"> - Jump - Mirror - Strobe - Teleport - Mini Doom 		

Done Newly Scheduled and done In Progress Removed



	Functional Minimum	Low Target	Desirable Target	High Target	Extra
Options	- none	- Single/Multiplayer - Volume	- Playmodes - Difficulty	- Customize Controller	
Achievements	- none	- none	- yes	- more!	- awesome.
Menu	- start	- functional	- animated	- awesome.	
HUD	- Distance Run (Numbers) - Distance to Doom - Runtime	- Graphs - Powerupstack	- Fixed to Player - Animated Stack		- Customize
Multiplayer	- none	- The farther you get	- Interaction through Powerups	- Same Track	- Online Multiplayer
Leveling	- none	- none	- Simple Basic Features	- Unlock "weapons"	
Misc	- none	- none	- none	- Contract with Blackrock or Disney ;)	- Contract with Blackrock AND Disney ;)

