Distant Souls

General Art Guidelines

General Art Guidelines

All art files must be uploaded in their original form (i.e. .png, .max, .blender) to keep a copy in case we need some sort of modification.

Modeling

All models face forward by looking towards positive Z. We use a real world scale, measured in meters. 1 engine unit = 1 meter.
The engine uses Ogre for rendering, therefore the files must be exported to ".Mesh" format. Fortunately there are exporters for all major modeling tools (Maya, 3DS Max, Blender)

Beware when having 2 faces looking in the opposite direction and sharing the same vertices, or having different vertices but being really close to each other. They may avoid the "hollow" look, but it plays really bad with shadowing. Only do this on parts that are needed or very notorious.

Also including a Blender-friendly format (Collada; obj, 3ds) is cool since I have a couple Blender scripts (in Python) to generate the Lua scripts used by the engine for physics representation based on the 3D shapes.

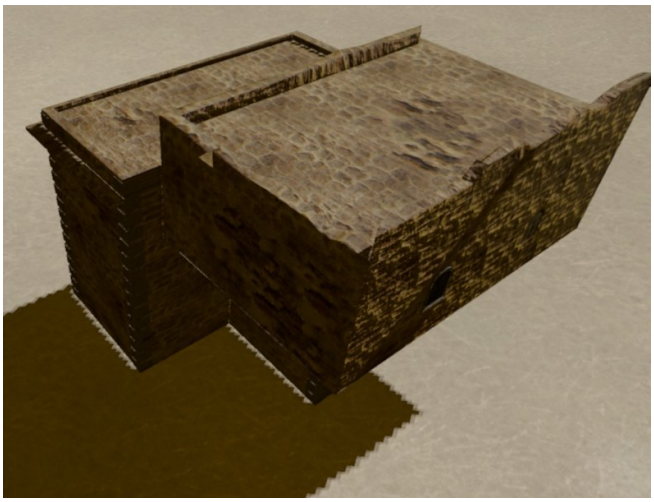
The engine has trouble with stairs. I'm working on it but it probably won't be fixed for the first act of the game. The way to fix this is by creating stairs with lots of steps and slightly offsetting each step so that it isn't really 90°, but rather a slight steep slope. When involving stairs, it takes exporting two or three times into the game until you get it right. There's an amazing technique to create stairs in no time by using edge loops and chamfer edges (3DS Max)/Bevel Center (Blender); so it's not really a problem, but something to keep in mind. See Appendix A on how to use this technique.

Most models should be centered at it's bounding box' center. In Blender, Object Mode, F9->Mesh Panel->Center New

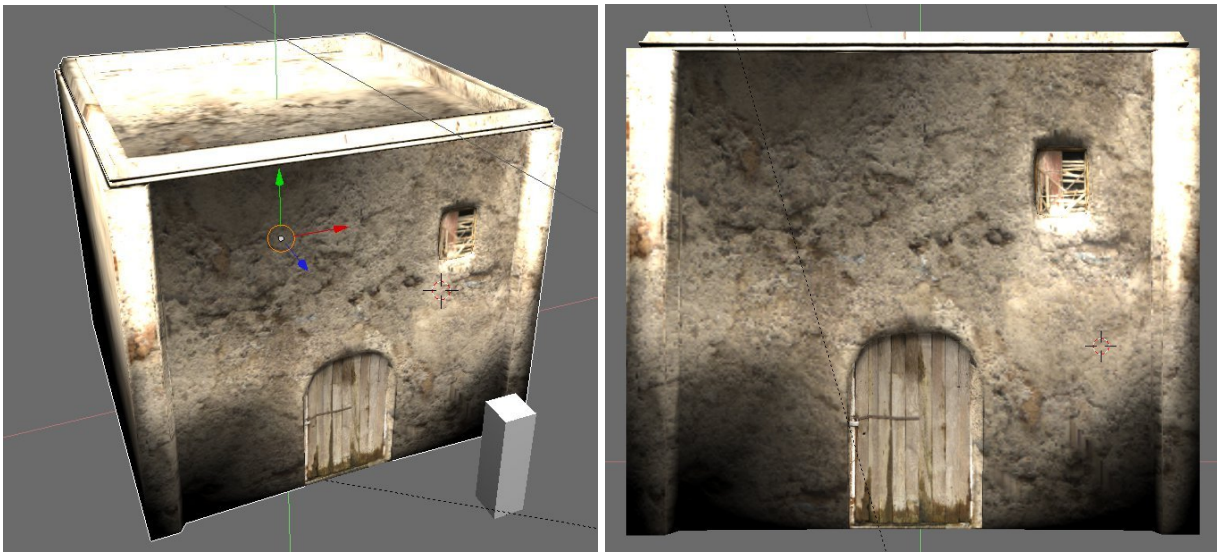
Buildings

Regular buildings should be around 400-2.000 vertices. Single mesh (single material)
If the building's form is concave, try to keep the vertex count lower, to play nice with the physics engine. If it's convex, the vertex count isn't relevant since a simple shape enclosing the mesh can be used (ie. A box) for the physics representation. Note compound physics shapes can be used in some cases (but sometimes is hard to make it work right). A compound object could be a box with a sphere on the top to represent a tree, for example.

Gameplay heavy buildings can be anything between 500-20.000 vertices. Can have multiple submeshes and materials. Textures are allowed up to 2048x2048 per material. Diffuse could exceptionally reach 4096x4096, but normal maps must always stay at 2048x2048 as maximum.

Example of a concave building. It can't be enclosed in a bounding box, since it would mean the ceiling would be at the same level (which isn't) and the characters wouldn't be able to walk underneath the building, as shown in the left picture.



Convex building being shown. In this case, there's a near perfect fit for a bounding box, using normal maps for faking complex geometry[1] while keeping the actual count to a minimum. This is excellent for less important buildings that need to be used for filling space in cities.

For example, Assassin's Creed(tm) titles make massive use of this technique. Lots of simple buildings, improved with some separate geometry details (a few doors and windows mainly), swapping the roofs for the same buildings, so that they don't look the same. To break the pattern and hide the fact they're using the same model over and over again, they're not always placed at the same level, **extra objects are added**, often gameplay related, such as *lamps, fire torchs, horizontal poles (sometimes the lamps are hanging from the poles), wood boxes piled together, pottery and similar objects, tables, markets, tents, etc*. These objects often allow to jump and climb them, allowing to reach the roofs. There's no reason we can't do the same. ***It's all in the illusion***.

The engine allows these objects to be completely movable rigid bodies, or just static, unmovable.

Main characters (humans)

These can have up to 6.000 vertices in total.

Head, body, hair and eyes must have different materials (different submeshes). Total number of submeshes/different materials = 4.

This obeys to technical requirements:

- The face can have facial animation, as well as having a separate texture, which results in higher quality heads.
- Hair has it's own material for cool shading, must be done separately.

1 The screenshot above was done sculpting a higher resolution model and then baking the normal maps

- Eyes also have their own material to take into account environmental reflections. This makes it feel more realistic.
- The body is well.... just the leftover. A generic shader with skeletal animation is applied to it.

The head's texture is allowed to use a resolution of 2048x2048

Ordinary characters (& human/non-human enemies):

Between 400-1500 vertices.
They'll be used for crowds, a few NPCs, & enemies.
They'll have a maximum of 3 different submeshes/materials. Ideally just one (the same material is applied to everything). This obeys performance factors: the less submeshes, the faster the game will run.

Art

Designs

Rounded shapes are usually linked with safety, calm; while spiky shapes are often linked with danger, strength.
We will follow that trend. NPCs will often be wearing rounded armors, shoulder pads should be rounded, pointing down; while enemies and strong allies should have their armour's shoulder pad pointing up or spiky. Dark colours, full of heavy items will denote stronger opponents.
This can be applied for environments. Places where the player should feel safe could be easily identifiable by it's round shapes and lighter colours.

Imitating Kingdom Hearts(tm) tradition for cloth styles[2], all armours and accessories don't need to have a big practical application, but rather **aesthetics will prevail over usability/functionality**. Just as long as the clothes or armour isn't absurdly useless and non functional.
Leather dresses for badass guys is preferred, with some extra metal parts.

Beauty of the girl (of course, what main characters' wish ; -)

The idea is to make her look attractive. Not a whore, but sexy. Not exhibitionist but not fully covered either. Subtlety (Ok the screenshot isn't exactly subtle). There's a fine line between looking sexy and being a bitch. We want them to feel sexy, something that you feel draw in. It doesn't necessarily mean the breasts are giant, but variety is probably best, depending on personality. Making them look slightly exposed, or in a tight dress, pointing up makes them more appealing. The point is, we're not aiming at 12 year old puberty kids where making boobs bigger with hardly any visible clothes means better.
Take a look, for example, at this excellent work “Open Green” from CG artist Crying Horn[3]:

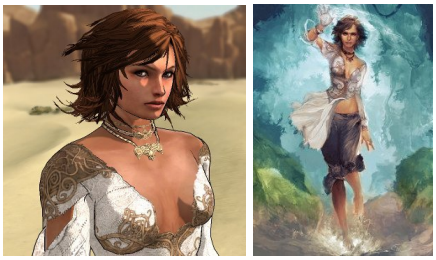
2 Disney and Square Enix are not part of the team nor involved in project

3 I do **NOT** own the image, and CryingHorn is not part of the team nor involved in project. **Used for explanatory purposes**. Believed to fall under fair use. Copyright © 2008 Andrius Balciunas

More of his work and an HD version of this image can be found here, here or here.

Look how she uncovers part of her back, yet she's not actually revealing anything, nor is being explicit. Note the face expressions and how she looks as if she's wearing make up. Her ornamental dress contributes to the design (the strings hanging falling from he shoulder, the bracelets, her long earring)

Another good example is Elika, from Prince of Persia 2008[4]:



Again, she's not naked, with the right of what needs to be shown (unlike the female companions from the previous 2 games). Her costume is also very detailed and stylish. Interestingly, her clothes also contains ornamental details.

Needed assets

Buildings

Characters (main and secondary, including generic enemies)

Big interactive castle

Small markets and tents

Water fountains

Arabian-like potteries to put in the streets

Debris, small details for environment

Dragons

4 I do **NOT** own the image, and Ubisoft is not part of the team nor involved in project. **Used for explanatory purposes**. Believed to fall under fair use. Copyright © 2008 Ubisoft

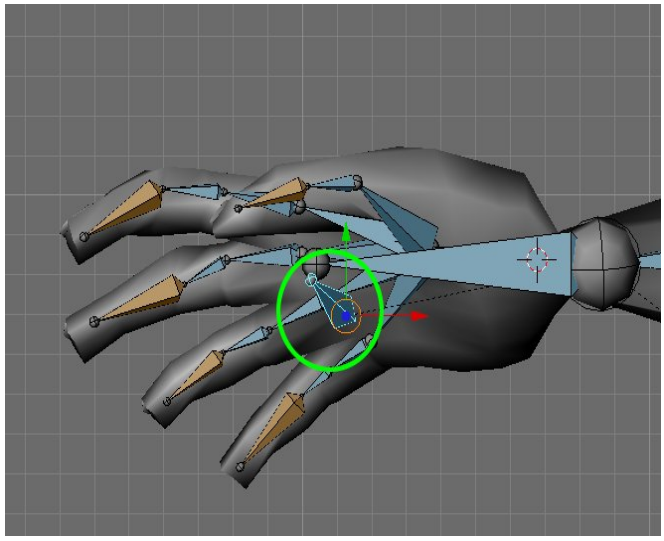
Animation

Rigging

When rigging, all vertices must have a weight of 1.0 with the bones. Failing to do so may result in shadowing glitches hard to detect. Also NVIDIA 3D Stereo Vision drivers don't like it (technical explanation here).
Using one bone per vertex is highly recommended, even for main characters. Cinematic characters may have up to 4 bones per vertex.

Bone count for a single object can not exceed 50 bones. Needless to say, keep it as low as possible for performance, but don't sacrifice quality on characters where the camera and player will be focusing a lot. If you really need to surpass the 50 bone limits, you'll need to consult me; as there are technical issues involved.

Bones can be used to put attachments. For example the current character model has 1 specia bone in each hand which isn't used for animating the vertices. Instead, weapons (or *any* object, actually) are attached to it. The orientation of the bone ***does*** matter, as it is used as a guide of how the attachments will be oriented.



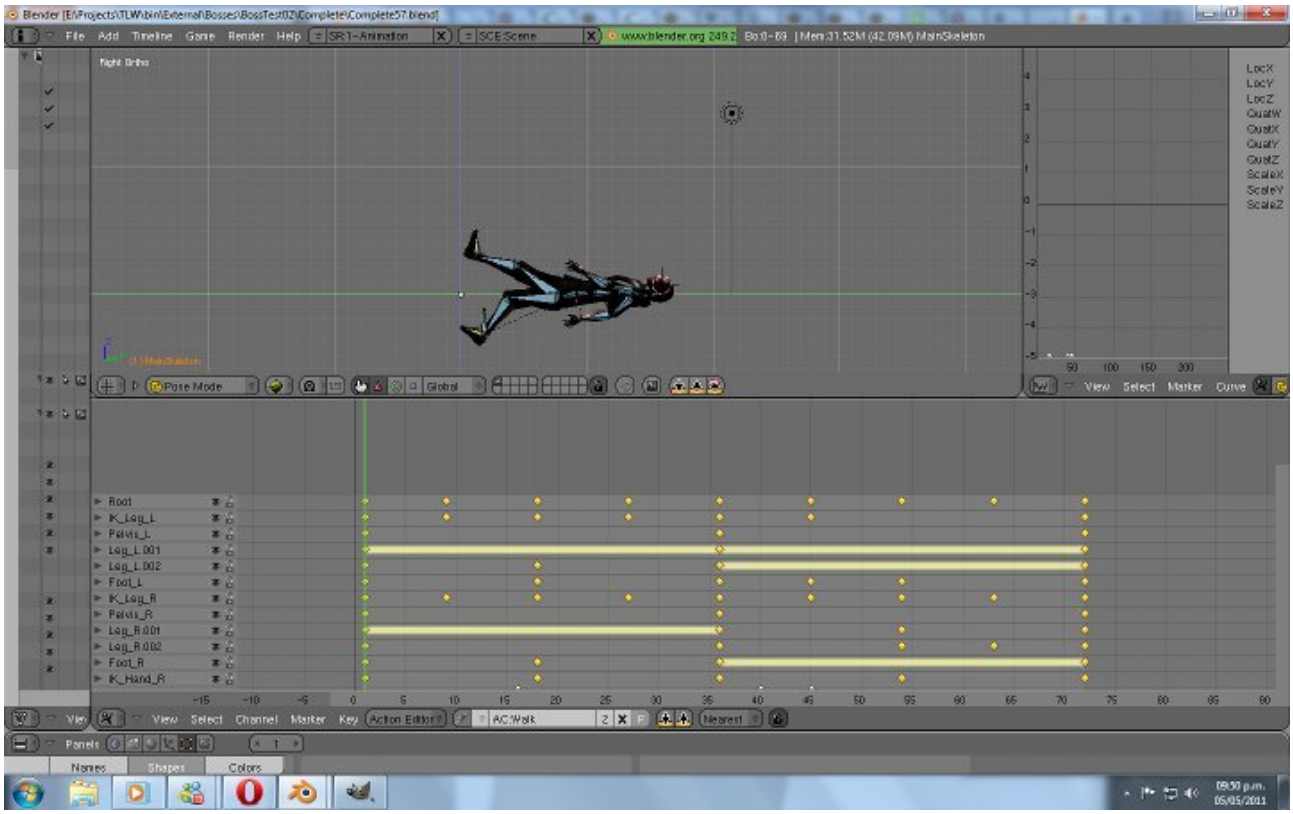
In the picture, the circled bone is used for weapon attachments. Note the bone is lose parented to another bone, that is, it doesn't move when it's parent does. This is not a requirement, but it helps with some attack moves that involve “throwing” the weapon or making it levitate around the user (like psychic powers)

Animating

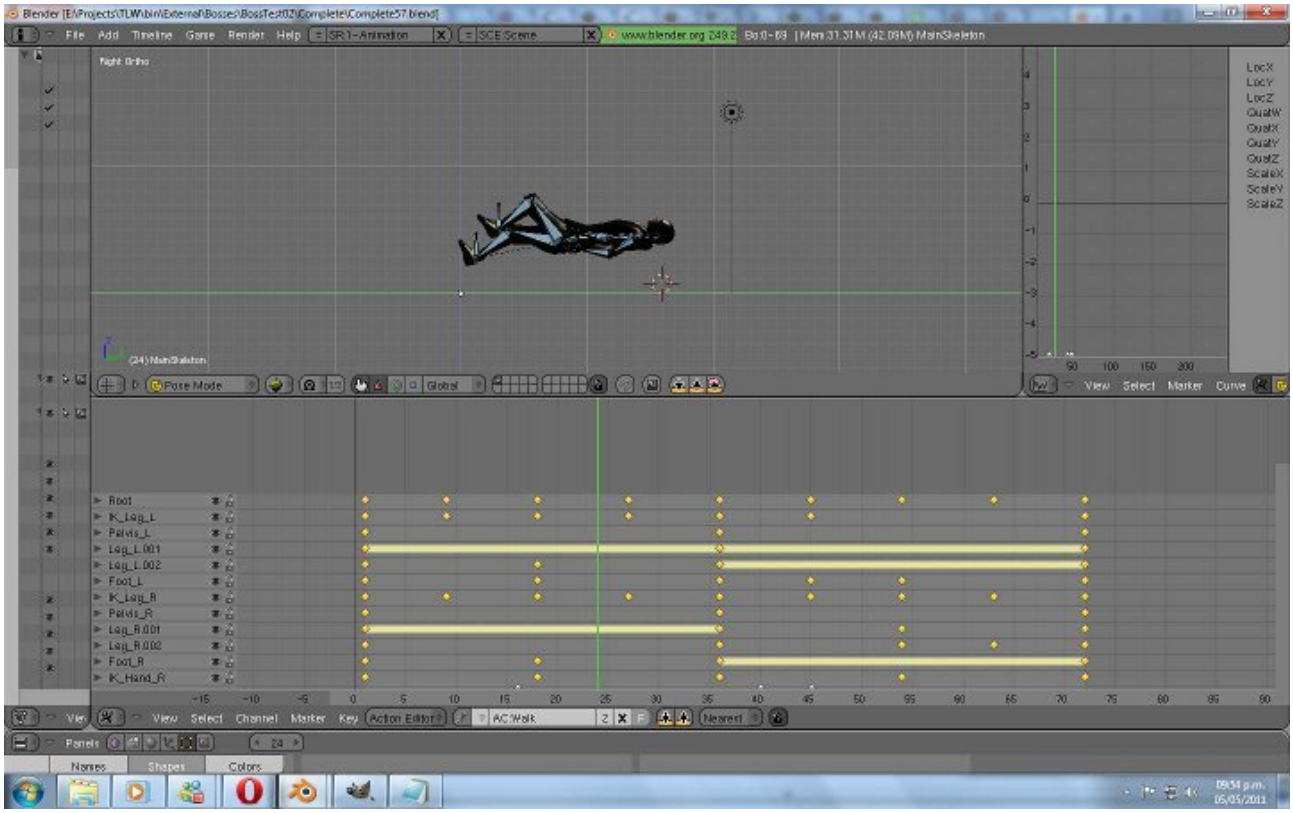
The engine is very flexible regarding movement and character animations. The motion speed isn't fixed to a single parameter, but rather is extracted from the model. This extracted velocity allows for a much more realistic movement, gets rid of the "slide effect" (the character walking looks like Michael Jackson sliding through floor) when done right, and allows lots of possibilities to the artist by the defining not only how it will move, but also how much it moves in each step, and where it moves at the specific keyframe (it doesn't necessarily have to be in straight Z axis, although this is recommended for the walk and run animations).
Note movement extraction in the Y axis has some limitations; however extraction may be disabled for that axis, which means the model will be animated but the in game representation won't move up or down following the animation (the in game representation is the key for physics collisions and receiving damage).

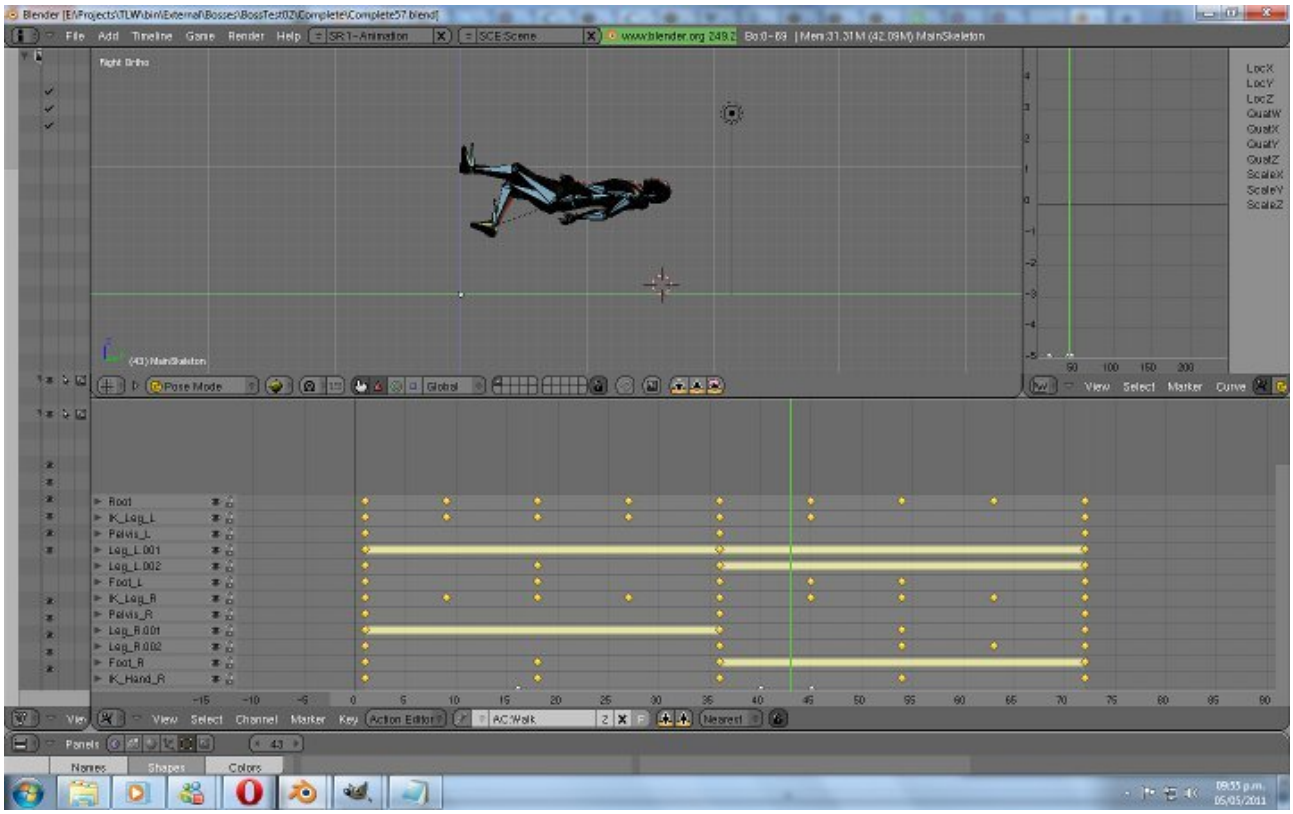
In other words, all this means that a walk animation doesn't loop on the spot like most game characters when watching it in 3DS Max/Blender/Maya. The animation has to actually move forward so the engine can calculate the distance traveled, automatically recenter the animation, and use the displacement as a velocity vector. Note the local movement should still loop (i.e. the arms, legs, etc from the last keyframe, relative to the object's center, should match the first keyframe's).

As an example, here's a walk animation in it's first keyframe:

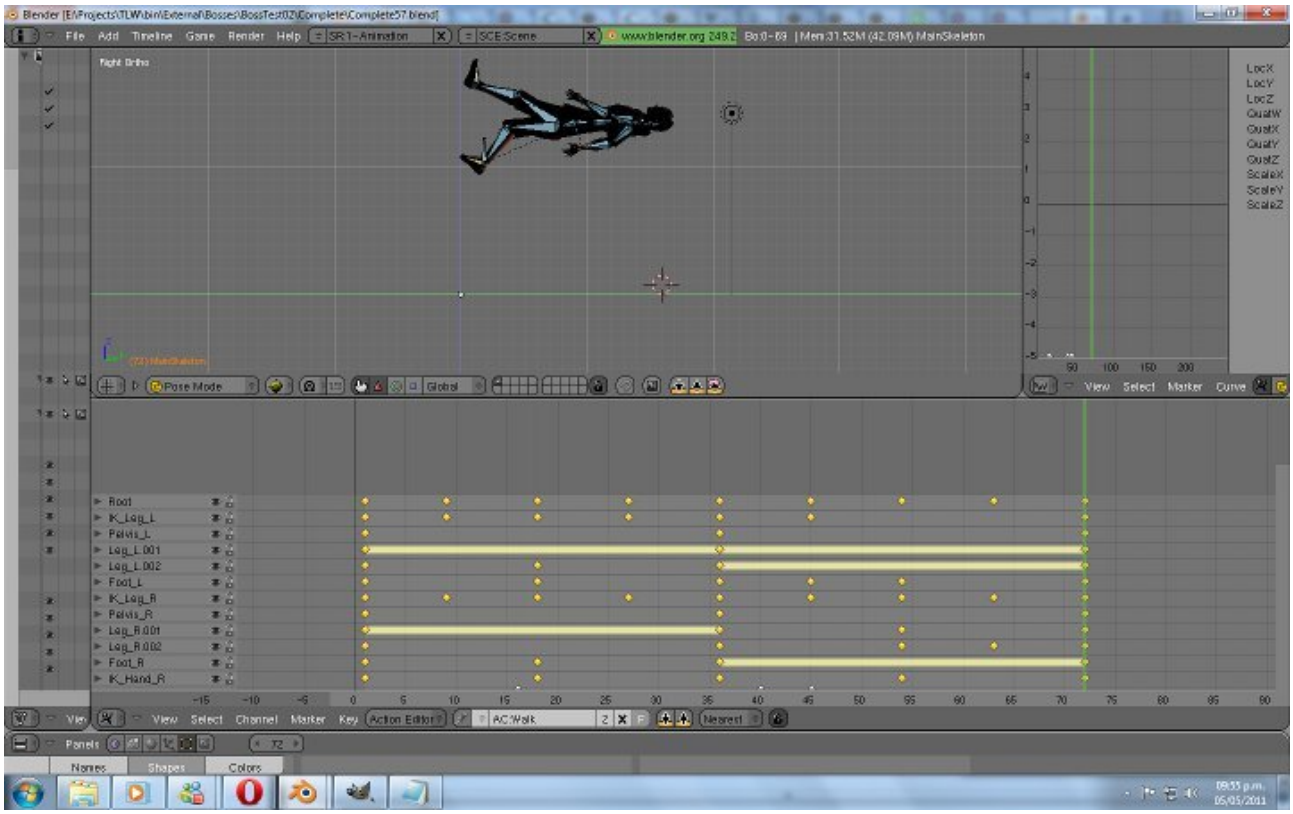


In the next keyframes, it also moves forward:





This is the last keyframe, note how the arms, legs, etc matches the position from the first keyframe, but with an offset:



This is a list of all animations that the main characters should have. Note this list may grow in the future. Other models may not need all of these:

Animation	***Description***
Idle	Self explanatory
Walk	Self explanatory
Run	Self explanatory, as a note, the steps frequency should be a proportional of that of the walk animation (i.e. if the walk animation takes 4 steps in 4 seconds, the run animation should make 4 in 2,6 seconds, but it shouldn't perform the first 2 steps in 1 second, and the other 2 steps in 4 seconds; unless the walk animation also does the same, proportionally). This isn't a requirement, but it helps the engine to blend between the walk and run animations seamlessly.
Jump	Jump on the spot (up)
JumpFwd	Same as Jump, but forward. Distance traveled is left to the artist's criteria, but must be gameplay tested. The animation has two stages. In the first stage it walks forward, in the second stage it actually jumps and becomes in air. The exact time where the stage is switched is set in a Lua script.
Air	Animation to play when the character is in the (falling) air and all jump animations have already ended.
FallHigh	When the character falls from a very high altitude, it crouches after touching the ground, recovering. The character is unable to move until recovered. 
FightStance	The “idle/ready” state when fighting and not performing any physical attack
MoveFwd	Moving forward when fighting (fight always locks direction towards the enemy). Note the animation is played in reverse when going backwards, be sure it doesn't look to bad.
MoveSideways	Moving sideways when fighting. Moving sideways can only be done while locked on something. (Zelda OoT style). The engine assumes it moves to the character's left, the same animation played in reverse is used to go right.
Attack01	Self explanatory. What's worth mentioning, attacks can be chained as

	combos. If attack01 combos into attack02, then attack01's final pose should match attack02's starting pose for seamless transition.
Attack02	Same as Attack01. Another interesting point, these animations may be used for anything that causes damage, for example magic
Attack03	
AttackN	There's no hard limit on how many attack animations can be done. This is to the discretion of the artist and gameplay designers.
Defend	Defend animation. While defending it can't attack, and it moves very slowly
DefendCounter01	Animation to play when the player countered an attack (hit the defend button with great timing)
DefendCounter02	Similar to DefendCounter01, but is played by the one whose attack has been countered.
HitFront	Hit / received damage from front (i.e. character bends back)
HitBack	Hit from back
HitLeft	
HitRight	
HitFrontCrit	Same as HitFront, but critical damage.
HitBackCrit	
HitLeftCrit	
HitRightCrit	
StepAsideLeft	Used by the AI. The player has a 3-member party. The other 2 party members will step aside when they run into the player's way. Makes them look clever. Nice detail 
StepAsideRight	Same as StepAsideLeft, but to the other side. Not necessarily a mirror of the other animation (gives more dynamism)
FusionPath	Special move animation. To be explained later, as it requires understanding the special move. The animation is no big deal anyway.
FusionFailure	Any kind of animation giving feedback that a cool move failed.


Particle effects were added in engine

Note all animation changes are automatically interpolated, and more than one may be active at the same time. The only animation transitions that aren't interpolated are the attacks.
How attacks chain into different combos is controlled via Lua scripts.

Bones names

Bones used for IK only (i.e. an IK target) must have the "_IK" prefix in their names. If the modeling tool allows it, set the bone not to deform geometry (i.e. in Blender, F9, Armature Bone panel, Deform button should be turned off). This way the exporter won't export the IK bone to the mesh, which means it won't count for the 50 bone limit; it also helps the in-game performance.

Attack naming conventions

Since combos can be linked together in any way; it's easier to keep track on them by using a special naming convention.
So far, I've came up with the following:
Attack_A_01
Attack_A_0103_02
Attack_A_02_03
The "A" means the combo. There may be up to 99 moves related to this combo[5]. The last number is the ID of this move. Number "0103_02" means this is the move ID = 02, and this move can start after the ending of either Attack_A_01 (ID = 01) or Attack_02_03 (ID = 03). This is because their ending poses matches the starting pose of Attack_0103_02.
In other words, after Attack_A_01 can only follow Attack_A_0103_02, and after it can only follow Attack_A_02_03; and after it Attack_A_0103_02 can be performed again if the combo definition allows it.

5 Not a hard limit by the engine. Just the naming convention.

Note: This naming convention is not mandatory, and you may come up with a better one. We're open to suggestions and improvements. The combo system is very flexible, meaning that it is even possible to create combos that change from “A” to “B” or making combos that never end.

Facial animation

Also called pose animation, shape keys, etc.
For technical reasons, no more than 4 shape keys, exceptionally up to 6, can be active at any time. This means that a face may have hundreds of shape keys, however at a single time, at least 96 of those keys must have a weight of 0.0
Later a shape may be turned off (w = 0.0) and enable a different one from those remaining 96.
I know, this is a very harsh limitation : (

Textures

By default all textures should be 1024x1024 or less; except where stated otherwise. Textures must be power of two. They don't have to be squared (i.e. 256x512 is allowed) but preferred.
Normal mapping is supported and encouraged. Displacement or parallax mapping is not supported and won't be.
Typically in game textures use DXT1 (which has high compression ratio) for diffuse, and U8V8 for normal maps. Use dds format for this.
Since DXT1 is a lossy format, a PNG backup of the diffuse texture must be kept.
If you don't know how to convert to DXT1/U8V8 formats, don't worry. It's very easy and I can help you.

Must read:

Here's a collection of useful external links. They're not really obligatory, but it will set you up in the vision for the game, or have been influential when making Distant Souls.

1. http://altdevblogaday.org/2011/03/26/whats-your-indie-strategy/
2. http://cmpmedia.vo.llnwd.net/o1/vault/gdc10/slides/Pangilinan_Erick_Uncharted2ArtDirection.pdf
3. http://cmpmedia.vo.llnwd.net/o1/vault/gdc10/slides/Lemarchand_Richard_Uncharted2_Postmortem_PLEASE-TURN-ON-COMMENTS-LAYER.pdf
4. http://archive.gamedev.net/columns/events/gdc2010/article.asp?id=1816
5. http://altdevblogaday.org/2011/03/17/built-here-no-thanks/
6. http://altdevblogaday.org/2011/04/24/indie-project-budgets/