CDestructibleBuilding

Step-by-step Pipeline V2.0 (May 11,2011)

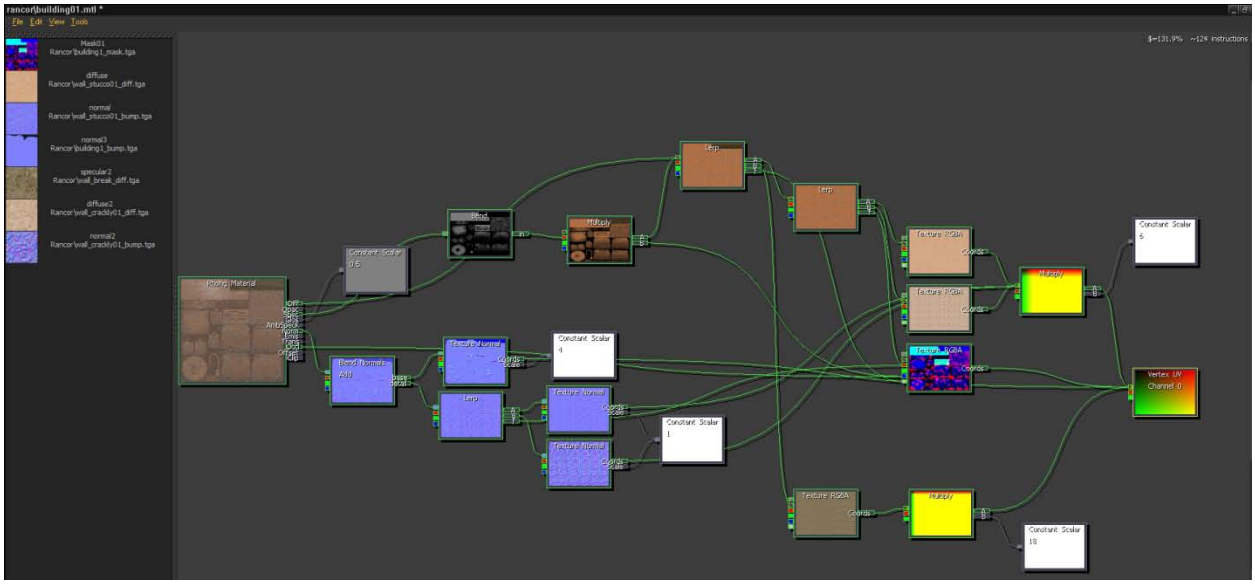
Final Intact Render from Infernal:

This document will walk you through the creation of the three basic sizes of buildings needed. They are referred to as small, medium, and large, which is based on the height of the building and not length or width. The doc goes through the creation a relatively cubical hut, but small buildings can have an L shape or be several hundred feet long and still be built in this same fashion. This doc will be focus on the small building with rule variations for the larger ones afterwards.

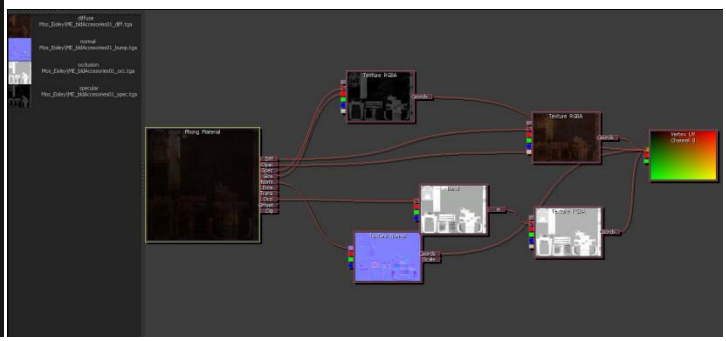
Part One: Basic Asset and Material Creation:

1. Model your building shell however you see fit. Optimizing your use of vertices is preferable for how assets will be rendered in game. The metal accessories and details are not to be combined with the actual building; they are handled as part of the destructible mesh and have their own material.
2. When unwrapping you will only need 1 uv channel. Mirroring is okay if it won't appear too obvious on screen. Save a strip at the top of your texture for a cracked texture for the inside edges once the model is chunked.

3. After you're uv's are laid out, you will need to export your intact state out as an obj. to import the model into 3dcoat (or similar program). Here you will need to sculpt in minor details like cracks, chips and other unique details that tiling textures can't do. When that is done you will need a black and white mask of where you want your two materials to blend together. In the example of the stucco/plaster buildings in mos eisley, its usually best to paint in or around the areas you sculpted cracks and chips with the normal map.
4. Both of these can be done in photoshop too, but painting directly on the model where you want the blend and normal details to be make the process much easier. After you are done painting the mask you will need to take that and put it in the red channel of your rgb mask. After that, you need to create another mask that is contained within the area you reserved for the chunk edges earlier. This will control what parts get a broken, chipped edge material. Then in your blue channel, you need to put in your AO map. If you take the blue channel of the normal map you painted earlier in 3dcoat and multiply it with your AO map, it will help make the normal map pop more and be more visible. Note: when making your AO make sure to bake your random pieces separately from your main intact model.
5. Once you are happy with your textures and masks, export your model into the Infernal Editor to begin creating the final material. Ideally your material will look something like this *Rancor/building01.mtl*. Try to make sure that the material is no more complex than the example and please make sure you select 'instanced' and 'simple' in the material geometry type. Most buildings should try to have the following texture allowances (less is always preferred if you can accomplish it without sacrificing quality).
    a. 4X 512x512 worth of standard textures
    b. 2X 512x512 worth of normal maps textures



6. For the accessories on my building, they need to be built separately and use their own texture and material; they do not need to be part of the intact shell. This allows you to make any accessories you wish, and incorporate them into any building you make.



Part Two: Creating the Destructible States:

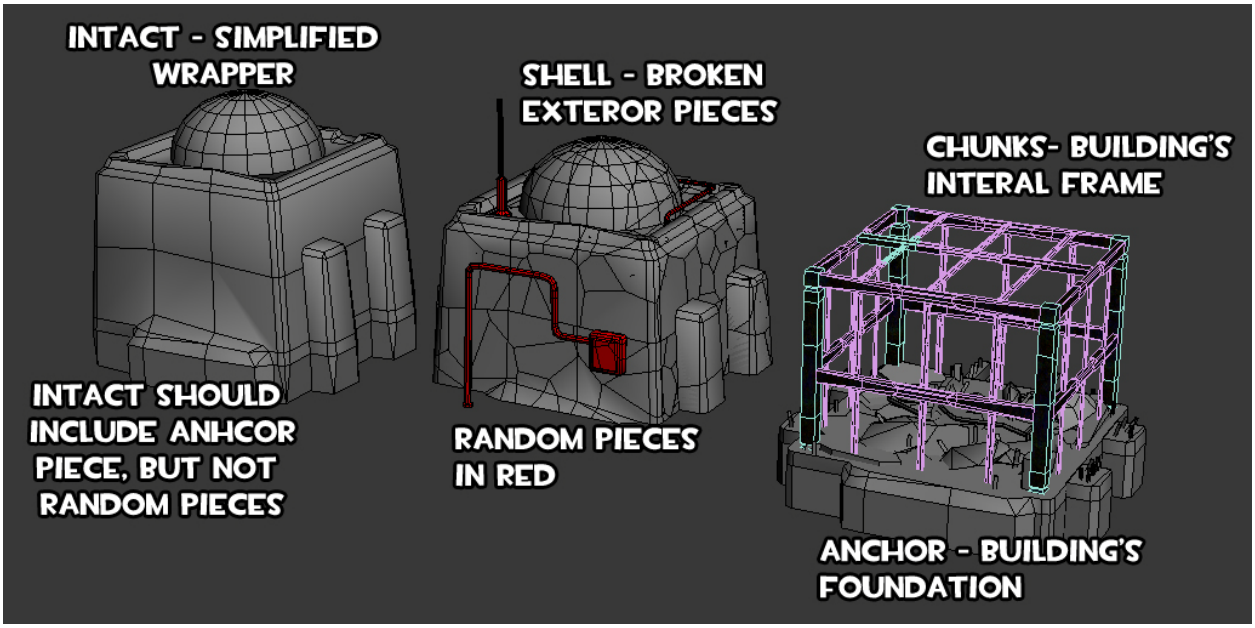
*It is highly recommended that you have some kind of cutting/capping program to aid in the creation of the breaking parts, this saves an immense amount of time and allows room for iteration. I used RayFire for this example, I would also recommend PullDownIt as an alternative.

* In terms of how the chunks and shell parts should be made, the program that can cut with a decent voroni or irregular pattern will be sufficient. Please try to make sure your pieces are approximately in the same size range. Avoid very small pieces.

*The beauty of our CDestructibleBuilding actor is the fact that the artist does not have to create ANY of the collision volumes for these buildings, they are programmatically generated! However, a very detailed naming convention must be followed in order to have the asset destroy correctly.

*The naming convention for Cdestructible works via a couple different layers. The first layer is your 'shell' layer. This is your external shell and represents the outer skin of brick, wood, metal, whatever primary material your building is constructed of. The second layer, called the 'chunk' layer, is generally going to be your primary building support. I-beams, rebar, enforced concrete, etc.

Taking these layers in order, here is how your broken pieces should be set up:



1. 'Intact' Layer
    - This is simply your intact model.
    - this part should include your anchor piece but not your random pieces.
2. 'Shell' Layer:
    - Part your intact model according to these principles:

✓ Try to keep your parts constrained to a building facing. We use compass directions to organize our parts. So, parts are arranged by my North facing wall, South facing wall, East, etc. The naming convention that you need looks like this: 'Shell_N01, Shell_N02, etc.'

- shell_N*
- shell_S*
- shell_E*
- shell_W*
- shell_T*

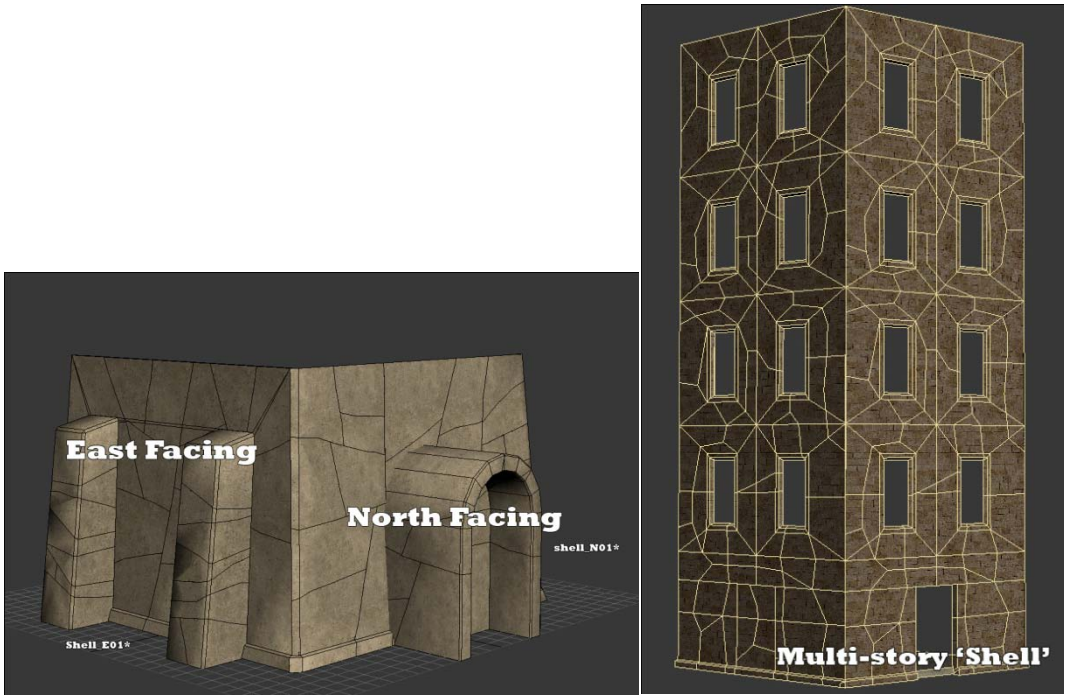
✓ walls should be about 1/2 ft (standard building) to 1 ft thick (more re-enforced buildings)

✓ For your shell fragments you want to have them be fairly small. A 20x20 to 25x30 ft section should have between 15-30 pieces.

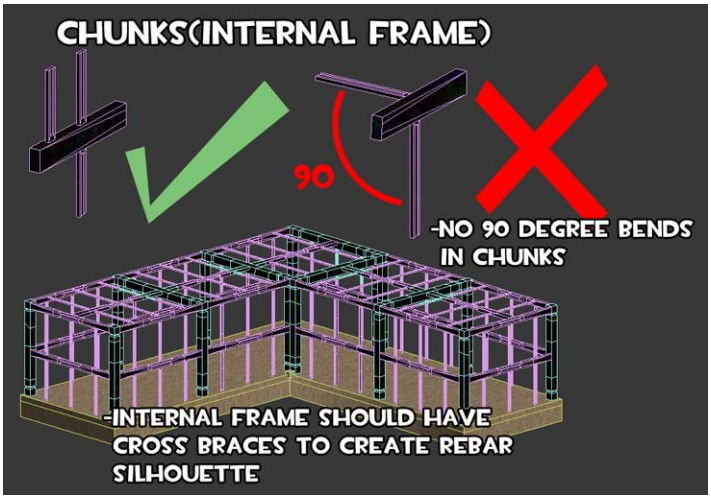
✓ A 20x20 to 30x30 ft section roofs should have fewer shell pieces in the 10-20 range.

✓ So when chunked, you will need to box map the edges of the chunks to the area you reserved and masked for the green channel. Its best to have a texture that tiles in all directions for this "cracked edge" texture since lining up uv's on chunked meshes can be very time consuming and in some cases with the rayfire tool, almost impossible.

✓ All pieces can only have one material assigned to them.



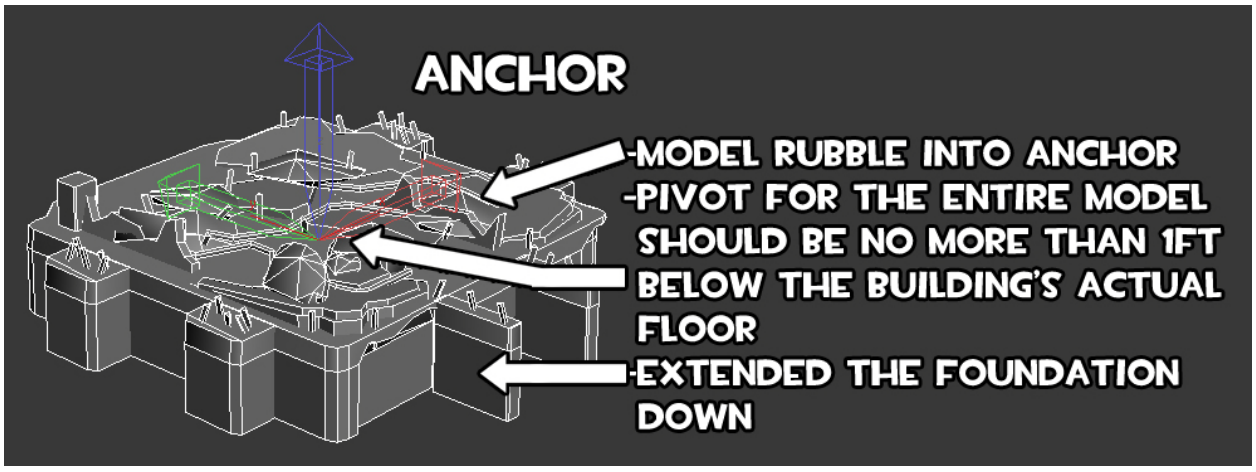
3. 'Chunk' Layer
    - Create your internal structure as you feel will best represent the structural components of your building and should include load and support beams.
        - ✓ Like 'shell' pieces, you can only have one assigned material to these pieces.
        - ✓ In general, your 'chunk' fragments will be of larger size than your 'shell'. A 20x20 to 25x30 ft section should have between 5-12 pieces.
        - ✓ For best results your chunk internal structure should sit flush with your shell pieces. At minimum you must have four support pillars per floor. They do not necessarily have to be at the corners but you need at least four contact points between a ceiling and a floor. for the



sake of visual interest, support beans will be need to link them together.

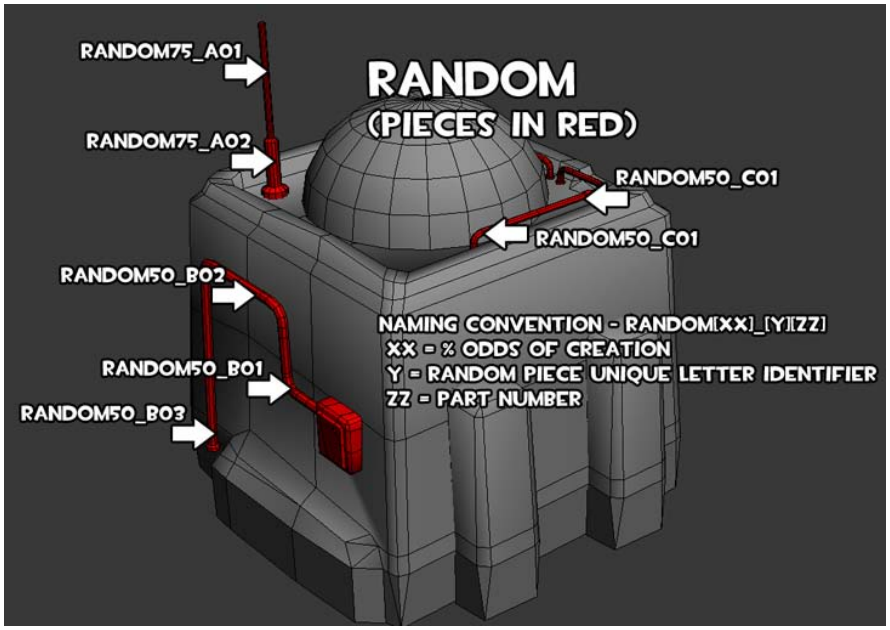
- ✓ All ‘chunk’ parts have a naming convention as follows: ‘chunk01, chunk02, etc.’
- ✓ You can build some representative damage into the inner structure of your building to further define the irregularity of the mesh parts.
- ✓ the chunks here should be relatively box shapped with their bounding boxes to prevent "floating" pieces. So pieces protruding off the chunks at angles should be avoided.
- ✓ All pieces can only have one material assigned to them.

4. ‘Anchor’ State
    - Your anchor state is any geometry you want to persist after your building has finished breaking. It needs to go beyond a simple slab to help sell the breaking. In order to help facilitate awesome looking building carcasses, you can use your anchor state for this. i.e. a rubble pile with a water pipe sticking out from the pile, etc.
        - ✓ Your naming convention is ‘anchor01, 02, etc.’
        - ✓ pivot to the model should be established here with the pivot being no more than 1 foot under the floor of the building(please use a pivot box)
        - ✓ extend the foundation down several feet to allow for any uneven terrain
        - ✓ All pieces can only have one material assigned to them.



5. Randomized Accessories

- We have the ability to generate random accessories on any building. How this works is you add any small details you want, pipes, A/C units, crenellations, windows, etc. to the breaking mesh. Depending on how you name the parts, at load, the actor will randomly determine which accessories to add to a building, creating a number of variations of the same building so that no two are the same, in theory.



  - ✓ The most import part of this feature is your naming convention. It follows this format: 'randomXX_YZ
    - XX = a value between 01-99 which is the probability that your item will be added to a building when it is created.
    - Y = the specific mesh part letter to group like parts together
    - Z = the number of the grouped part
    - EX: random15_A01, 02, random25_B01, 02, etc.'
  - ✓ All pieces can only have one material assigned to them.
  - ✓ since random parts can range in size it is hard to put a limit on the number of pieces. But since there are mainly relegated to details it should always be very few 1-10 (see image).
  - ✓ Random parts do not need to be represented in your intact state.

6. Buildings variations

Any building under 25 feet tall should be built with the above guidelines. So the shell wall section on a building that is 60 feet long and 25 feet tall should be made up of no more than 50 - 60 pieces. However that ratio of piece to feet shrink the higher the building goes. In general it is better to error on the side of fewer pieces, but don't do it at the sacrifice of the overall quality of the break. We should generally avoid really big, long, or tall buildings.

- 22 feet high 20-30 foot section of wall
    - 20-30 shell pieces
    - 5-12 chunk pieces
- 22-44 feet high 30 foot section of wall
    - 10-25 shell pieces
    - 4-10 chunk pieces
- 44-75 feet high 30 foot section of wall
    - 5-15 shell pieces
    - 4-8 chunk pieces

With any luck, if you followed this guide, your building should come in and be ready to be destroyed! Please refer to the attached max file and screenshots to help you better understand this process as it applies to a real asset

Building File pieces

modelName = rancor\\mos_eisley\\building01.smf

- location of model that the building file is referencing

useSupportStructureSim = 0

- 0 = building can support itself under all damage [default, this is what most building should be]
- 1 = building will collapse under its own weight after several of its chunk supports have been destroyed [this is for larger buildings]

destructionLevel = 1

- type of sim used - 0 = player can walk through[default - most buildings should use this level, save the higher numbers for larger or re-enforced buildings]
- 1= player can walk through the shell but must punch through the chunk frame
- 2=player must punch through the shell and chunk frame

lightHitFxFilename = rancor_punch_concrete_light.tfa

- FX played when rancor hits the building with a light punch

heavyHitFxFilename = rancor_punch_concrete_heavy.tfa

- FX played when rancor hits the building with a heavy punch

snapFxFilename = rancor_building_joint_break.tfa

- FX played when the connection between pieces is broken

rollFxFilename = rancor_tat_chunk_roll.tfa

- FX played when piece impacts the set with force

monsterDamageSndEventName = play_rampage_break_stone

- sound played when building is damaged

impactSetSndEventName = play_rampage_imp_sand

- sound played when piece hits set

impactSndEventName = play_rampage_imp_stone

- sound played when piece is hit by something else