Pre-computing Lighting in Games

David Larsson

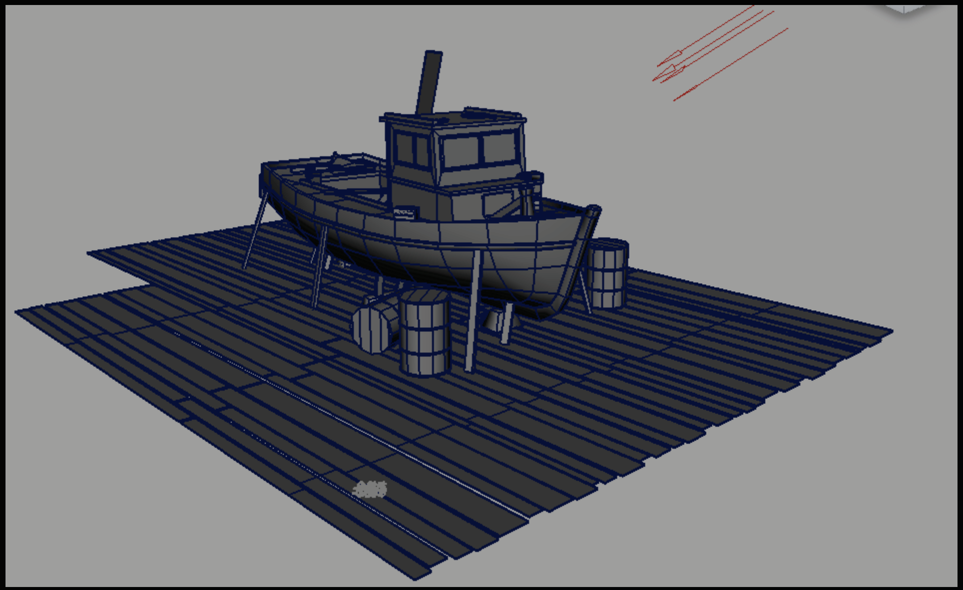
Autodesk Inc.

Autodesk

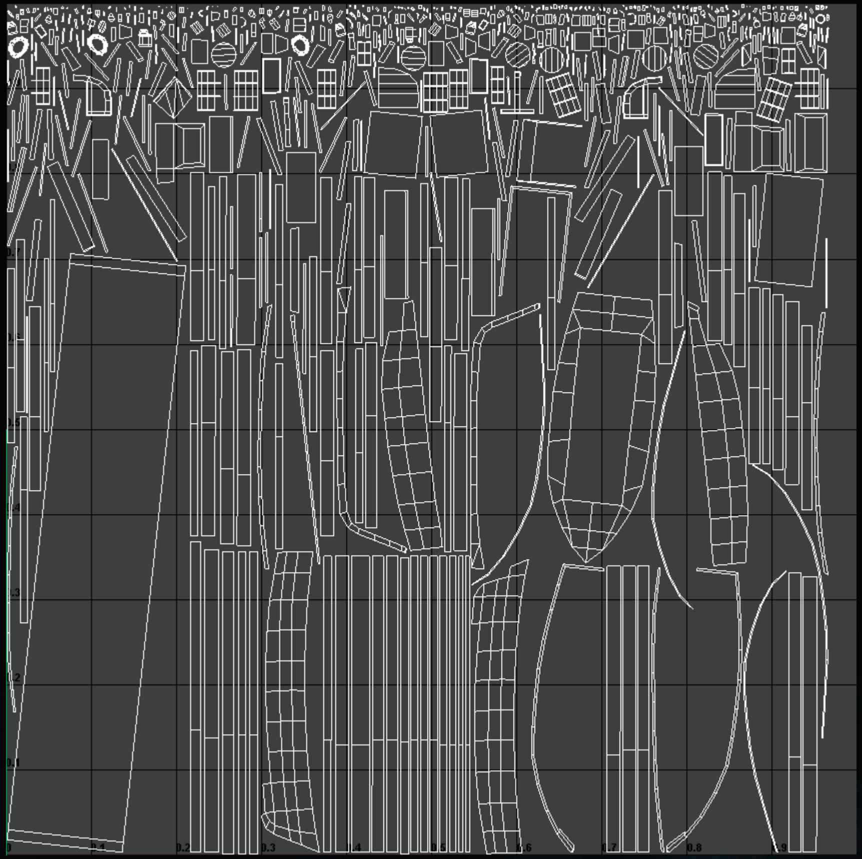
What is baked lighting?

- Precompute lighting information for static scenes and lights
- Typically baked to
  - Vertices
  - Textures
  - Light probe points in space for relighting of dynamic objects
- Most common approach to get access to GI in games
- Independent of GI algorithm
  - As long as it works in texture space

What is baked lighting?



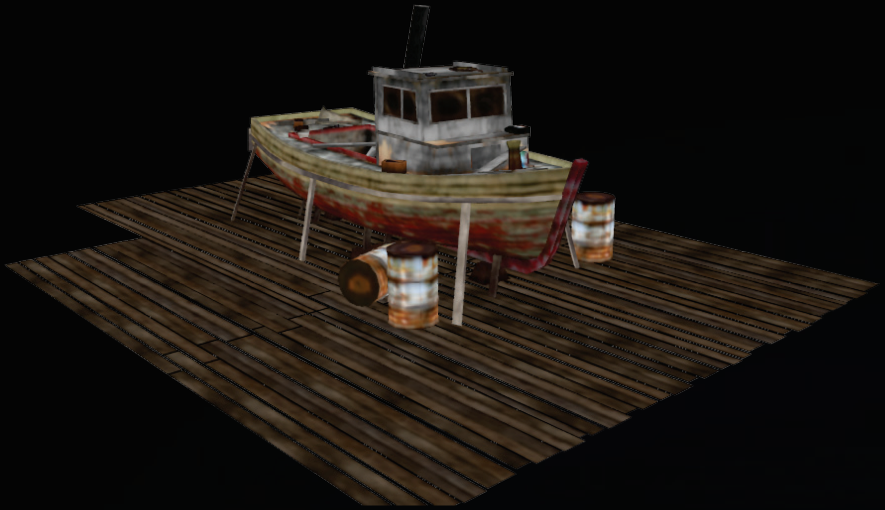
Wireframe scene



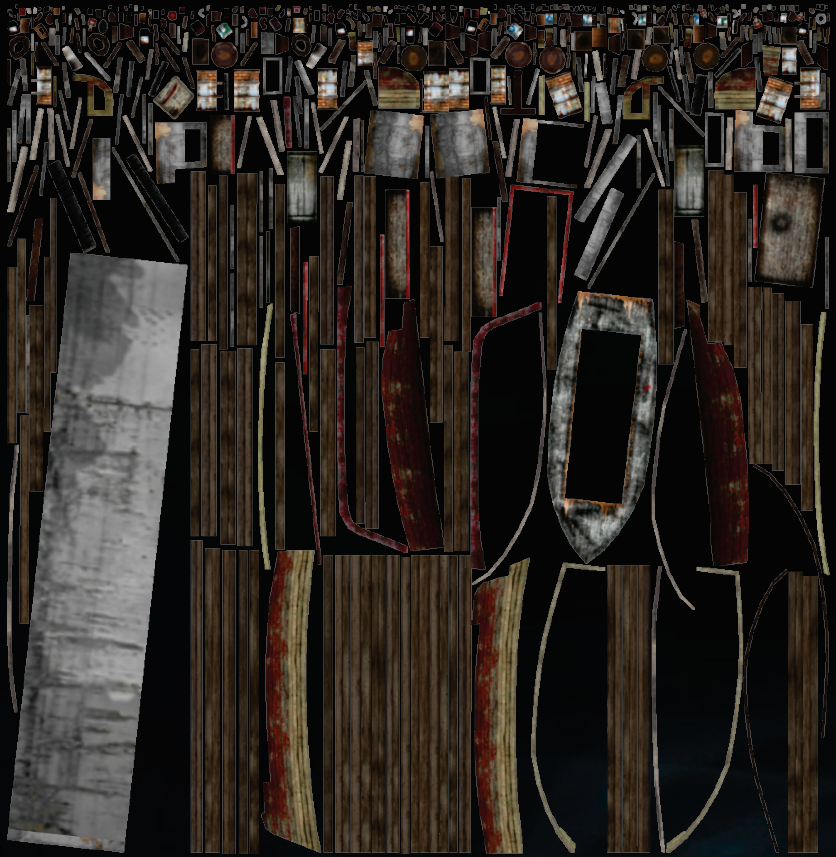
Wireframe UV layout

Content courtesy of A2M

What is baked lighting?



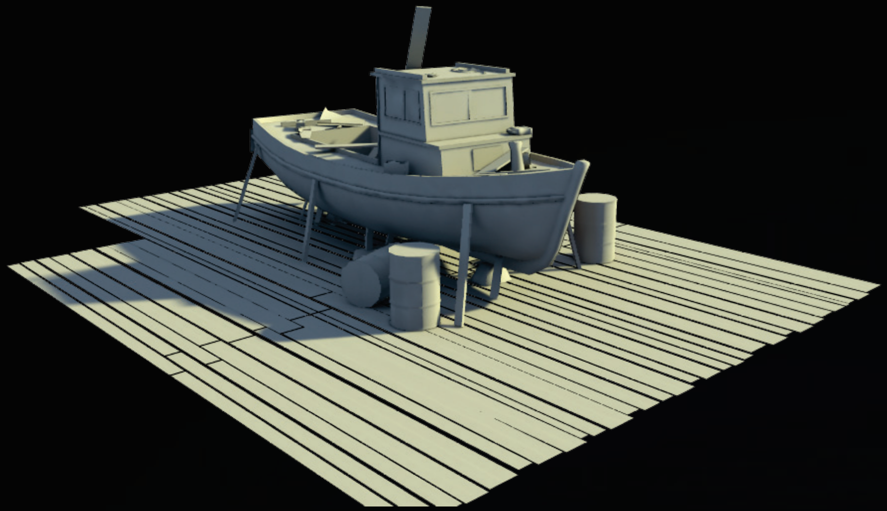
Diffuse Reflectance scene



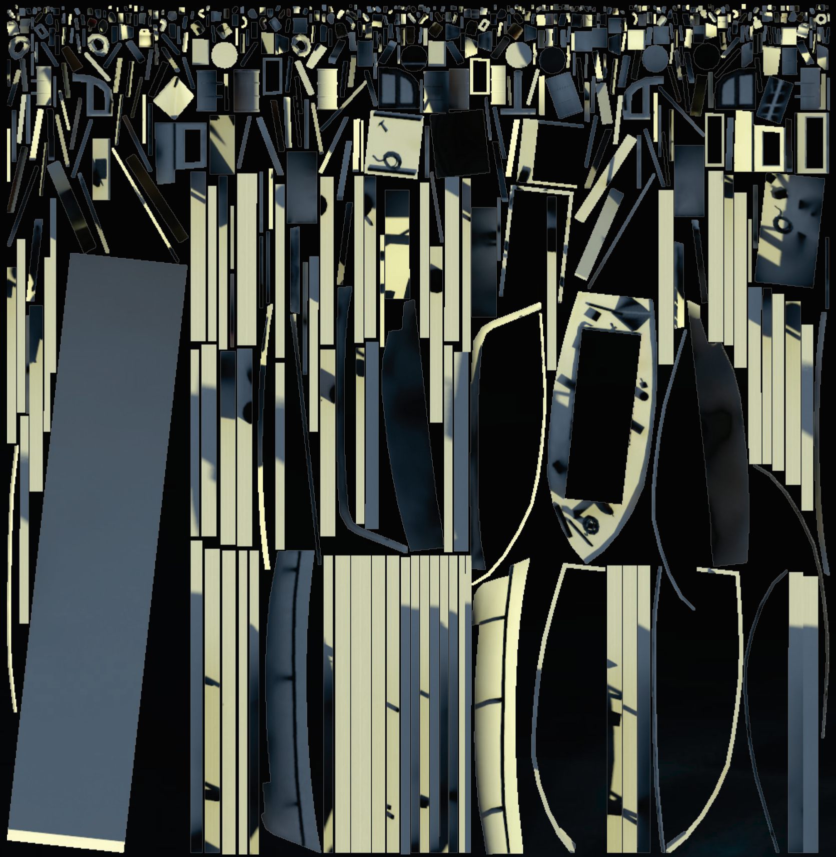
Diffuse Reflectance in UV space

Content courtesy of A2M

What is baked lighting?



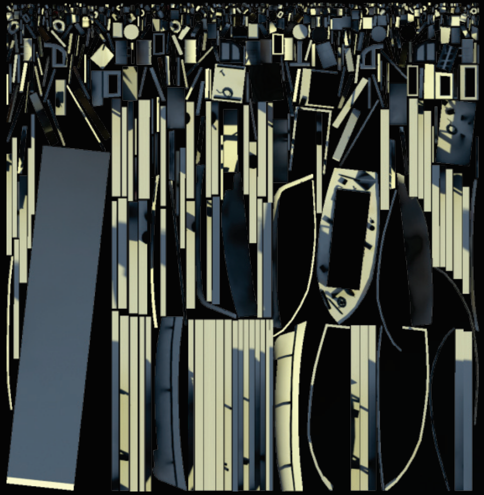
Lighting only Scene



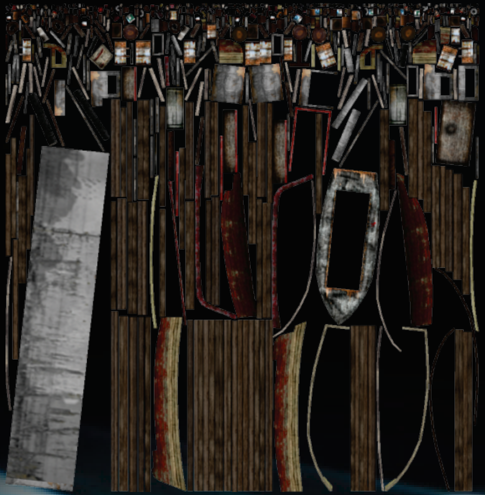
Lighting only in UV space

Content courtesy of A2M

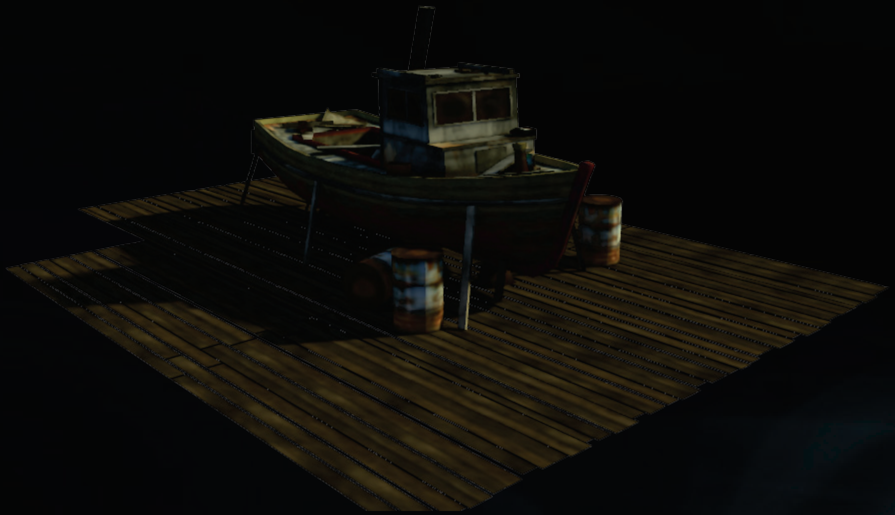
What is baked lighting?



*



=



Content courtesy of A2M

Why bake lighting?

- Quality
- Performance
- Lighting Workflow

Quality

- Allows the highest quality light simulation algorithms
  - GI effects
  - Multiple bounces
  - Allows high quality direct lighting

Quality

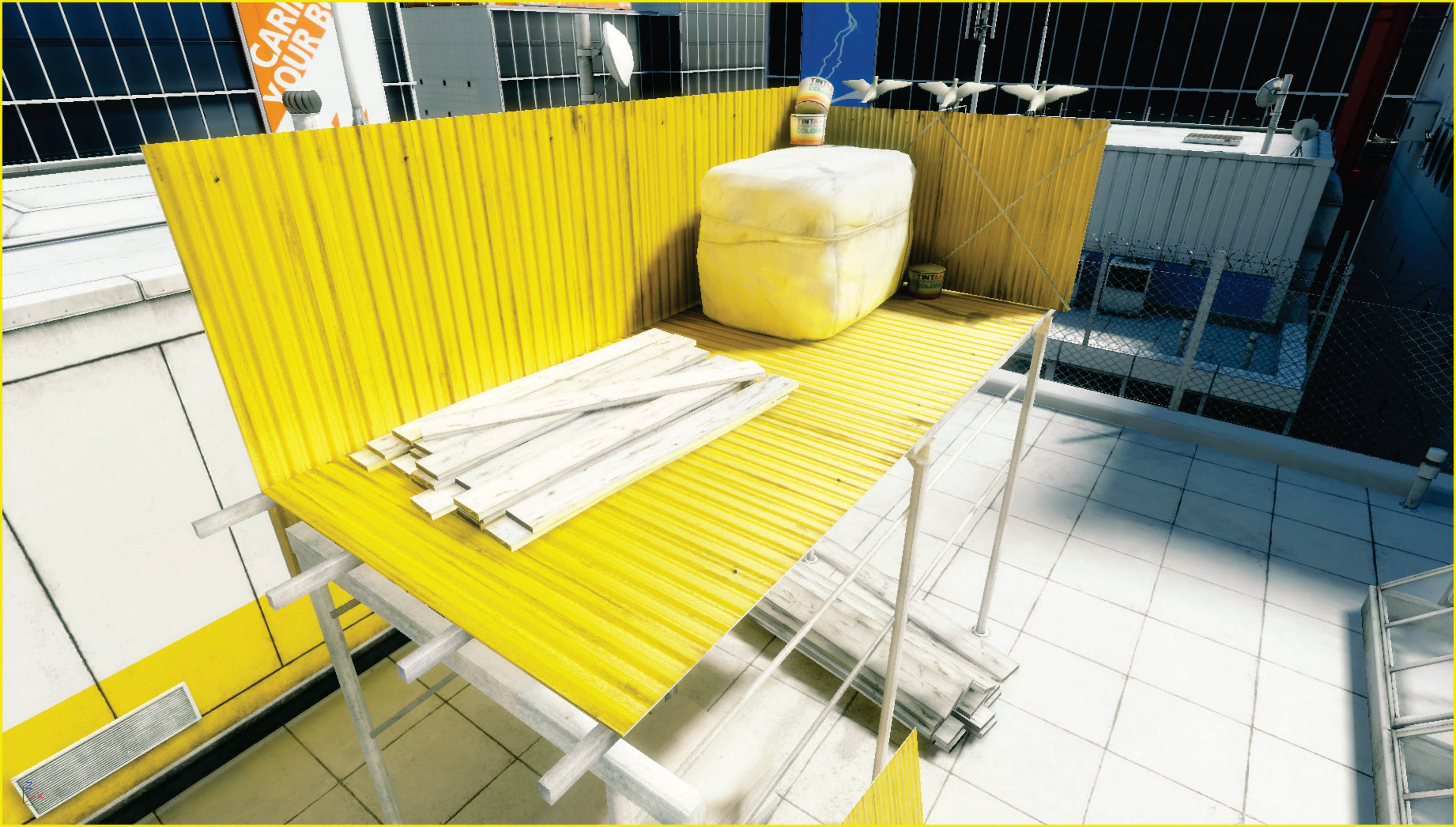


Image from Mirrors Edge, by EA DICE

Workflow

- Baked lighting is a way to give artists access to Global Illumination (GI)
  - Define the lighting in terms of actual light sources
  - No artificial fill lights
  - Decouples lighting from the geometry/materials
- Baked Lighting allows a richer set of light sources
  - Physically based Soft Shadows
  - Shadow Casting HDR light probes

Performance

- Runtime performance very good
- Independent of light setup
- Independent of GI algorithm
- Good looking light maps run with the same performance as poor looking ones

Predictable Performance

- Runtime performance tends to be very robust
- Artists can add as many lights as they want
- Realtime shadow map performance and GI less predictable
  - Light angle and position affects the performance of the shadow rendering
  - Player position affects what lights needs resolution

Scalable Performance

- Worked in Quake 1
- Used on handheld devices today
- Used in today's high end games

Challenges

- Changing Light Setups
- Moving/Deformed Geometry
- Memory Usage for the Baked Lighting
- Light Rebuild Times

Memory Usage

- Lighting is global
- Material textures
  - Instances share material textures
  - Multiple objects can share textures
  - Textures can be tiled and mirrored
- Lighting textures
  - Must be unique per instance
  - Cannot be tiled, mirrored etc
  - Possible to optimize resolution based on resolution requirements
- Reference on memory usage for lightmaps
  - Lightmap Compression in HALO 3, Hu

Normal Maps and Light Maps

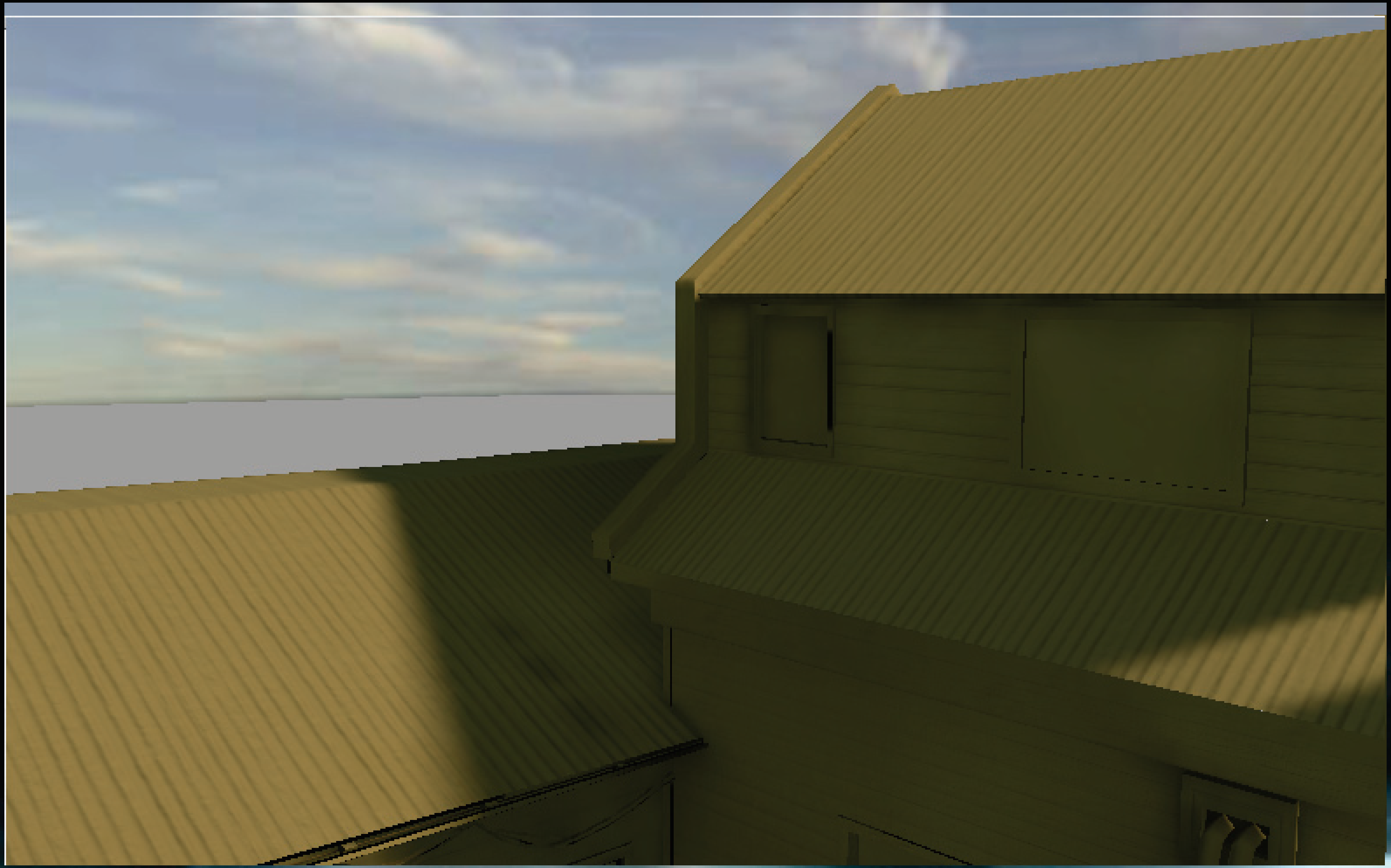
- Normal maps are great for increase the geometry detail level
- Normal maps introduces high frequency details in the lighting
- High frequency lighting requires high texture resolution

Directional Light Maps

- Details are in the geometry, not in the incoming lighting
- Store the hemisphere of incoming light per texel in the light map
- Allows approximation of lighting for different normal directions

Directional Light Maps

Content courtesy of A2M



Low resolution Directional Light Maps combined with normal maps

Baked Lighting

Normal Maps

Directional Light Maps

- Typical Encodings
  - Radiosity Normal Maps (RNM)
  - SH (generally 2 bands, 4 components)
  - Per pixel ambient and directional light
  - H-basis
- Allows real BRDF:s
  - Hemisphere is blurry but it’s possible to get reasonable specular effects from it too
- References
  - Half-Life 2 / Valve Source Shading, Gary McTaggart
  - An Efficient Representation for Irradiance Environment Maps, Ramamoorthi et al
  - Efficient Irradiance Normal Mapping, Habel, et al

Changing Light Setups

- Possible to bake different times of days
  - Combinatorial explosion if introducing more changeable lights
- Treat moving and intensity changing lights like ordinary runtime lights
  - Good for explosions or lights that flicker
  - No indirect lighting

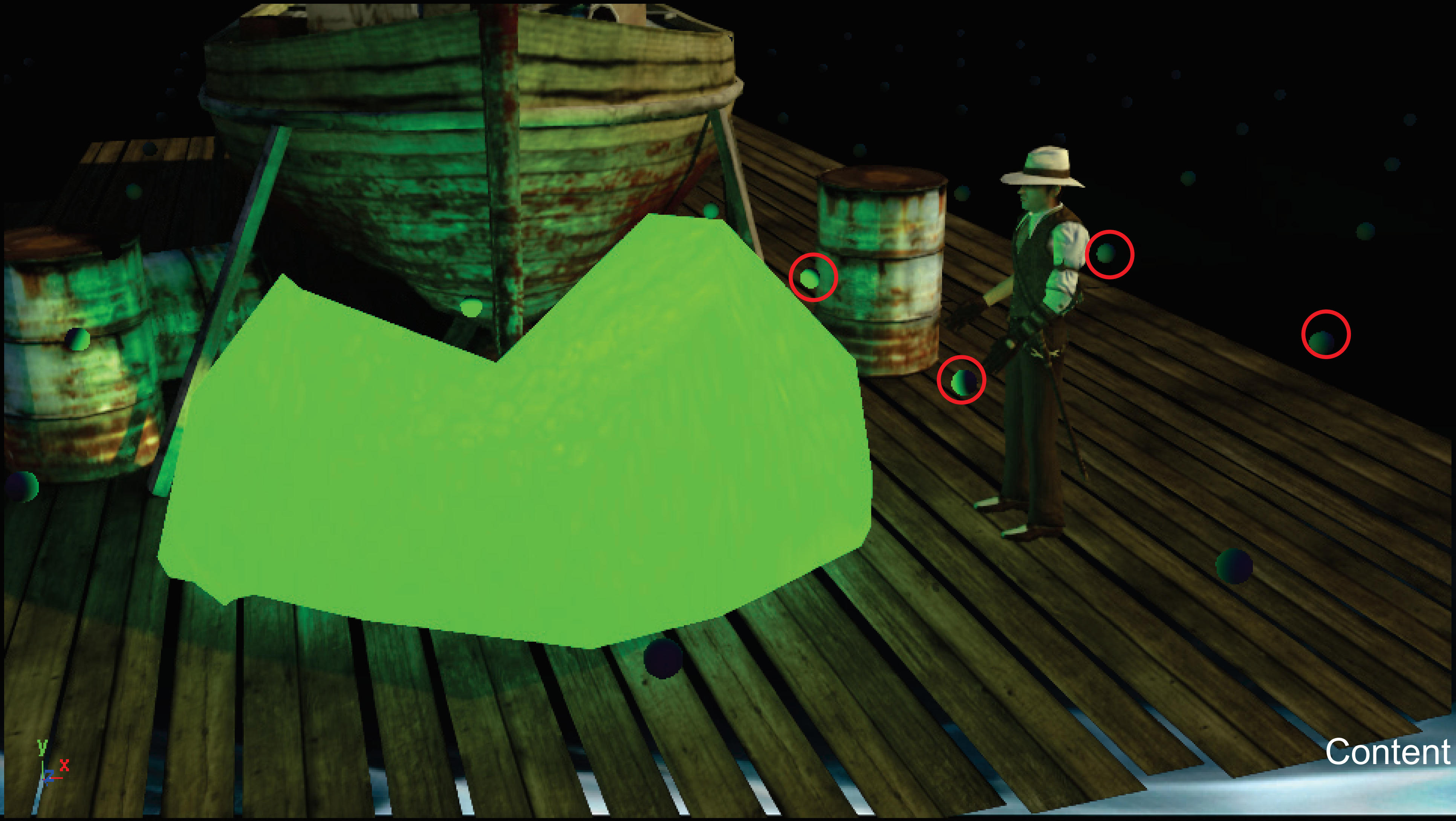
Changing Geometry

- Separate between local and global changes
- Local
  - Characters moving in a room
  - Small furniture
  - Bullet holes
- Global
  - Destroyed buildings
  - Destroyed walls

Local Geometry Changes

- Two sides of the problem
  - How is the object affected by the environment?
  - How does the object affect its environment?

Incoming Light on Moving Objects



Content courtesy A2M

Autodesk

Incoming Light on Moving Objects

- Bake light probes in the room
- Use the closest ones to light the object
  - Approximate the incoming lighting as one light probe for an entire object
  - Works well on objects small compared to the environment
  - Very large objects may need special treatment
- Encodings
  - Spherical harmonics (typically 3 bands)
  - Cube map with 1 pixel per side
  - Single ambient color

Moving Objects Affecting the Environment

- Direct lighting lighting in light probes optional
  - Allows self shadowing on dynamic objects
  - Allows the object to cast shadow on the environment
- Possible to extract strongest light direction from light probes too
  - Described in [Stupid SH tricks, Peter Pike Sloan]
  - Gives the possibility for self shadowing from indirect lighting
- Some titles only bake indirect light for lights where character shadow on environment is a big deal
- Indirect lighting from characters generally insignificant

Global Geometry Changes

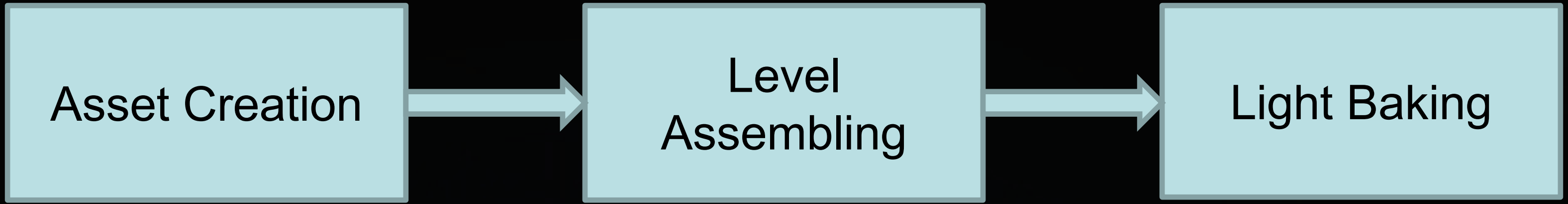
- Highly dynamic games tend to avoid global baked lighting
- Other subsystems tends to rely on or perform better on static geometry as well
  - Path Finding
  - Collision Detection
  - Game story often requires players following certain paths

Hybrid Solutions

- Bake only indirect lighting
  - Indirect light generally smoother than direct lighting
  - Sharp shadows needs higher texture resolution
- Special treatment of the sun
  - Sunlight is often the most influential light for outdoor scenes
  - Direct sunlight often a source of sharp shadows and dynamic range differences
  - Bake indirect only from the sun, add direct as a runtime light

Pipeline

Level and lighting workflow

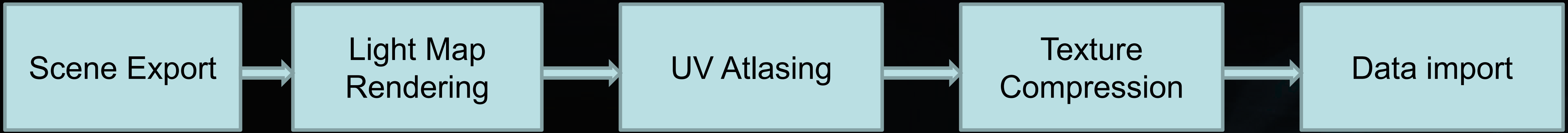


Asset Creation

Level Assembling

Light Baking

Breakdown of the Light Baking phase



Scene Export

Light Map Rendering

UV Atlasing

Texture Compression

Data import

Pipeline implications

- Light build stage can be time consuming
  - In the magnitude of CPU hours
  - Dependent on algorithm, resolutions, level size, light setup, number of bounces etc
- Tools to speed things up
  - Selective Light Builds
  - Preview Quality Builds
  - Preview Tools
    - Camera render tools
    - Progressive light map generation
  - Distribution
- Automatic rebuilds to make sure lighting is always up to date

Pipeline Implications

- Clear separation of what is static and dynamic
  - Both for Lights and Geometry
- Tools for placing and managing light probes in levels
  - Grids
  - Hierarchical grids
  - Arbitrary points
- Tools for managing texture resolutions and bake type

Pipeline Implications

- Tools for managing GI specific light source properties
  - Scale factors for direct and indirect lighting in order to exaggerate and separate light contributions
- Tools for managing GI specific material properties
  - What gives good glow effects and the right look on screen is not necessarily giving the desired light emission on the environment
  - Increase or decrease overall reflectivity for scenes

Pipeline Implications

- Texture baked shapes needs unique UV
  - Possible to automate to some extent
  - Content that is easy to unwrap is preferable
  - Keep details in the normal map layer if possible
- Vertex baking is common
  - No seams because of insufficient texture resolution
  - Normal maps together with directional light maps can help give details in low resolution lighting
  - Not good with shadows and other lighting discontinuities inside polygons

Questions?