Cubes All The Way Down

FEZ Technical Post-Mortem

By Renaud Bédard, Programmer (Polytron)

About Me

- Started messing with 3D programming in VB6 & Truevision3D around 2001
- Started working with Phil Fish on FEZ in April 2007
- FEZ gets 2 nominations and 1 award (Visual Arts) at IGF'08
- Bacc. in Computer Science at UQÀM in late 2008
- Worked full-time since then at Polytron to make the game
- FEZ is first commercial title and full-time “industry” “job”

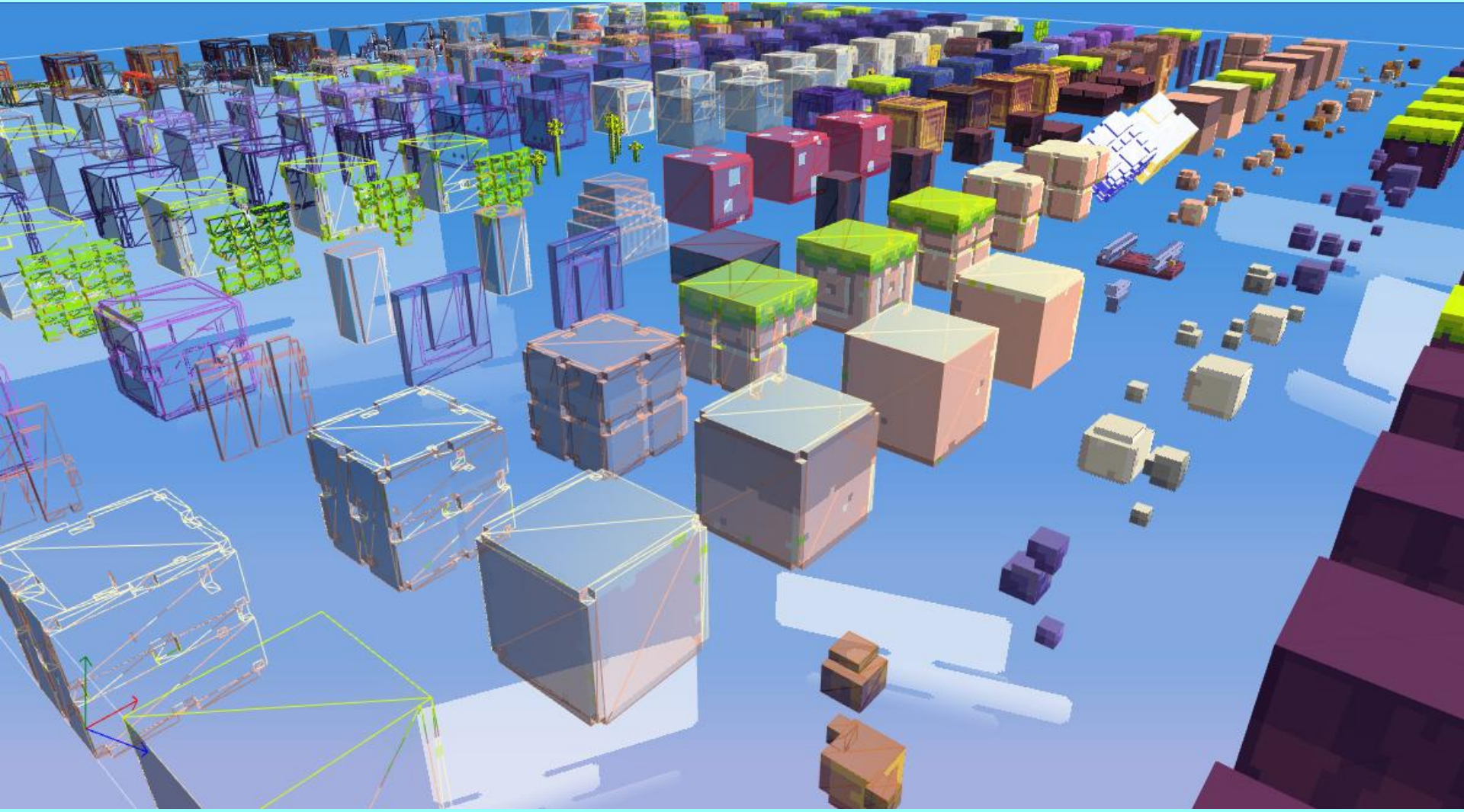
- 2D/3D Exploration Puzzle Platformer (“mystroidvania”)
- Pixel art where the pixels are 3D *trixels*
- Platforming in 2D, but across all 4 orthographic views
- Built in XNA/C# from the start
  - Spanned 5 XNA versions, from 1.0 to 3.1!

Long Screenshot

How is the world "cut up"?

- Art and level production-wise, easier to work with tiles
- Plain old 2D tilemap doesn't work : we need 3D tiles
  - ...*triles*?
- 16x16x16 trixels chosen as an arbitrary fixed trile size

Nature Tile Set (302 tiles)



Texturing Triles

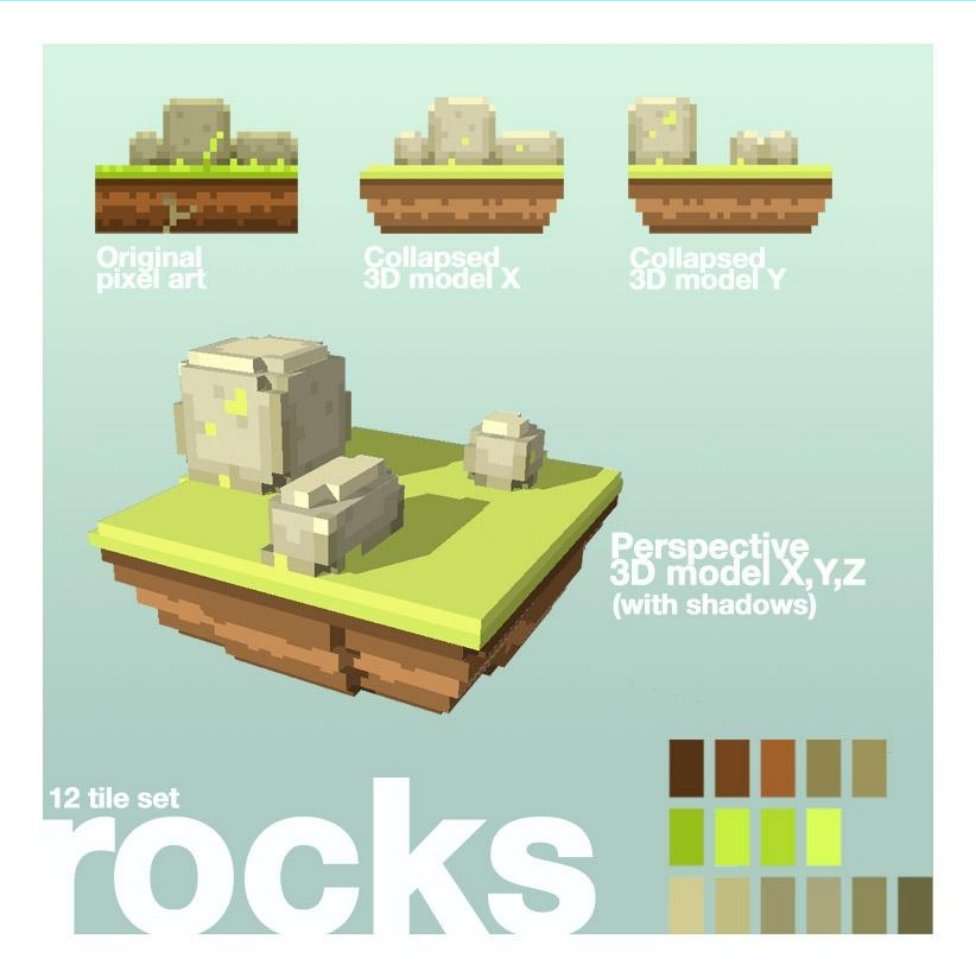
- Per-trixel coloring?
  - Implies good painting tools in modeler

- Cubemaps! (freebloods) 

  - Triles need to be convex
  - Cubemaps are 16x16x6 32bpp = 6Kb uncompressed!

Modeling Trile

- First trile concepts made in Google Sketchup

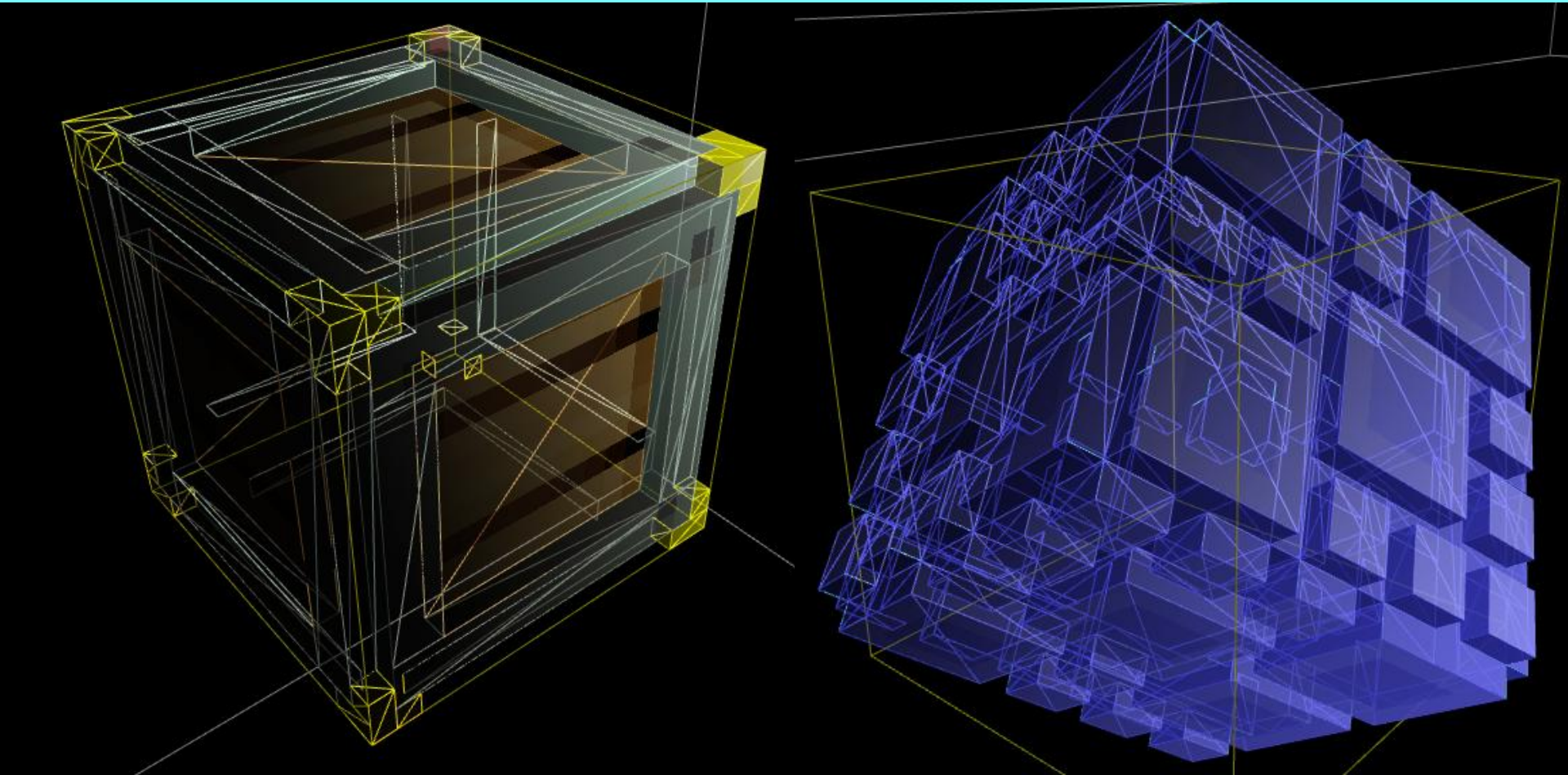


Modeling Trixels

- Trixel sculpting
  - Push trixel faces in, pull faces out
  - Adaptative geometric complexity

Modeling Tiles

- And things can get pretty complex

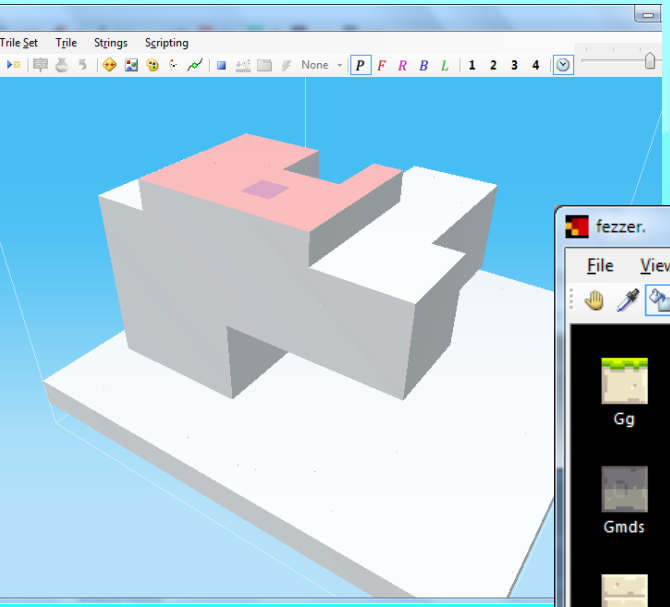


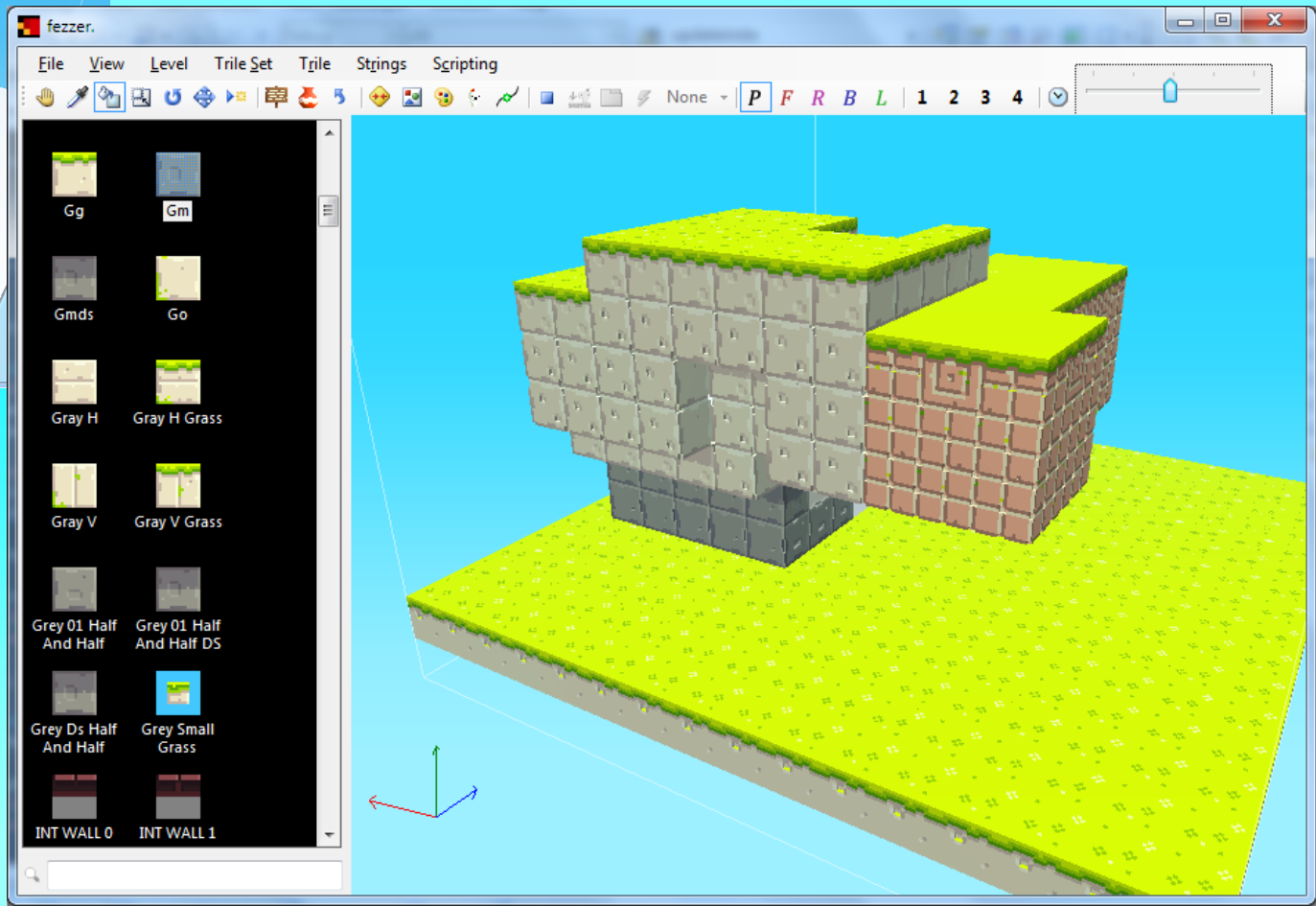
Mesh Simplification

- Extrapolate contiguous surfaces from trixels
- For each surface,
  - Enumerate biggest rectangles in that surface
  - Each rectangle becomes a plane

- Triles have an average of 136 polygons
- Typical scene : 50,000 polygons on-screen
  - Goes from anywhere from 5000 to 250,000

Building Levels in "Fezzer"





Rendering Triles : Culling

- Orthographic views
  - Peel the frontmost layer
  - Keep offset and see-through triles
- During rotation transitions
  - Peel the two visible faces
- Always use frustum culling
  - Only cull the difference, leave what's still visible there

Culling Disabled

Culling Enabled

Rendering Trile : Instancing

- Lots of instances of the same trile in a level
- Few things specific to a single instance
  - XYZ Position, Y Rotation : 4 floats

- vfetch Instancing on Xbox 360
  ≈ SM 3.0 Hardware Instancing on PC

- Max. 226 instances of same trile per draw call

Instancing shader details

- Instance data passed as VS constants

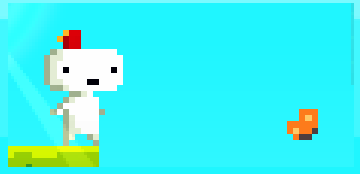
```
#define InstancesPerBatch 226
float4 InstancePositionPhiArray[InstancesPerBatch] : register(c30);
```

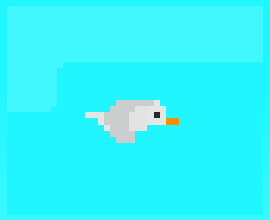
- Reconstruct transformation matrix in VS

```
float sinPhi, cosPhi;
sincos(InstancePositionPhi.w, sinPhi, cosPhi);
float4x4 instanceMatrix =
{
    cosPhi,         0,        -sinPhi,          0,
    0,              1,        0,                0,
    sinPhi,         0,        cosPhi,           0,
    InstancePositionPhi.xyz,                    1
};
```

Other World Objects : Planes/Decals

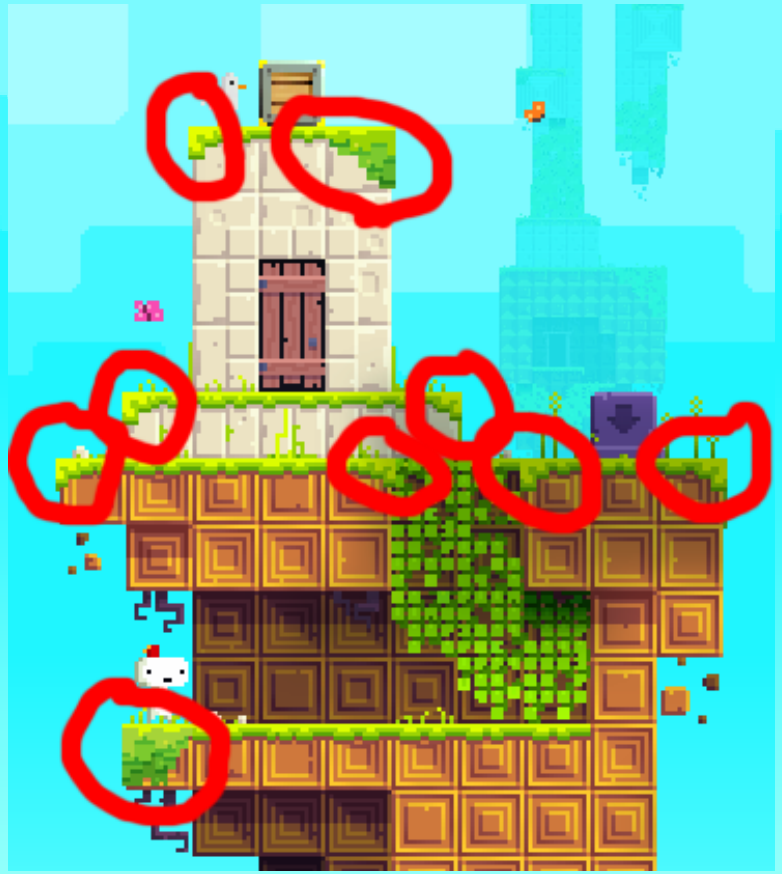
- Where sculpting doesn't matter, or sprite animations





Other World Objects : Art Objects

- For small overlapping details, or unique bigger landmarks



Collision Management

- Triles *are* the collision map

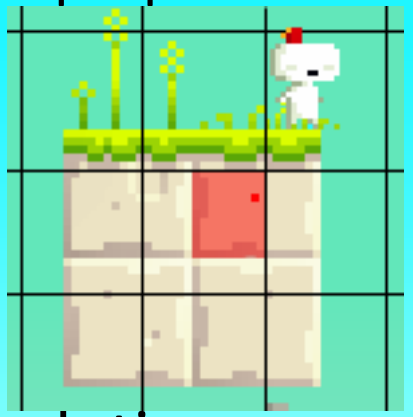
- Four types
  - Immaterial (e.g. blades of grass)
  - No collision (background elements)
  - Top-collide (most platforms)
  - All-collide (blocking boundaries)

Invisible Collision Triles

- Art objects are filled with invisible collision triles

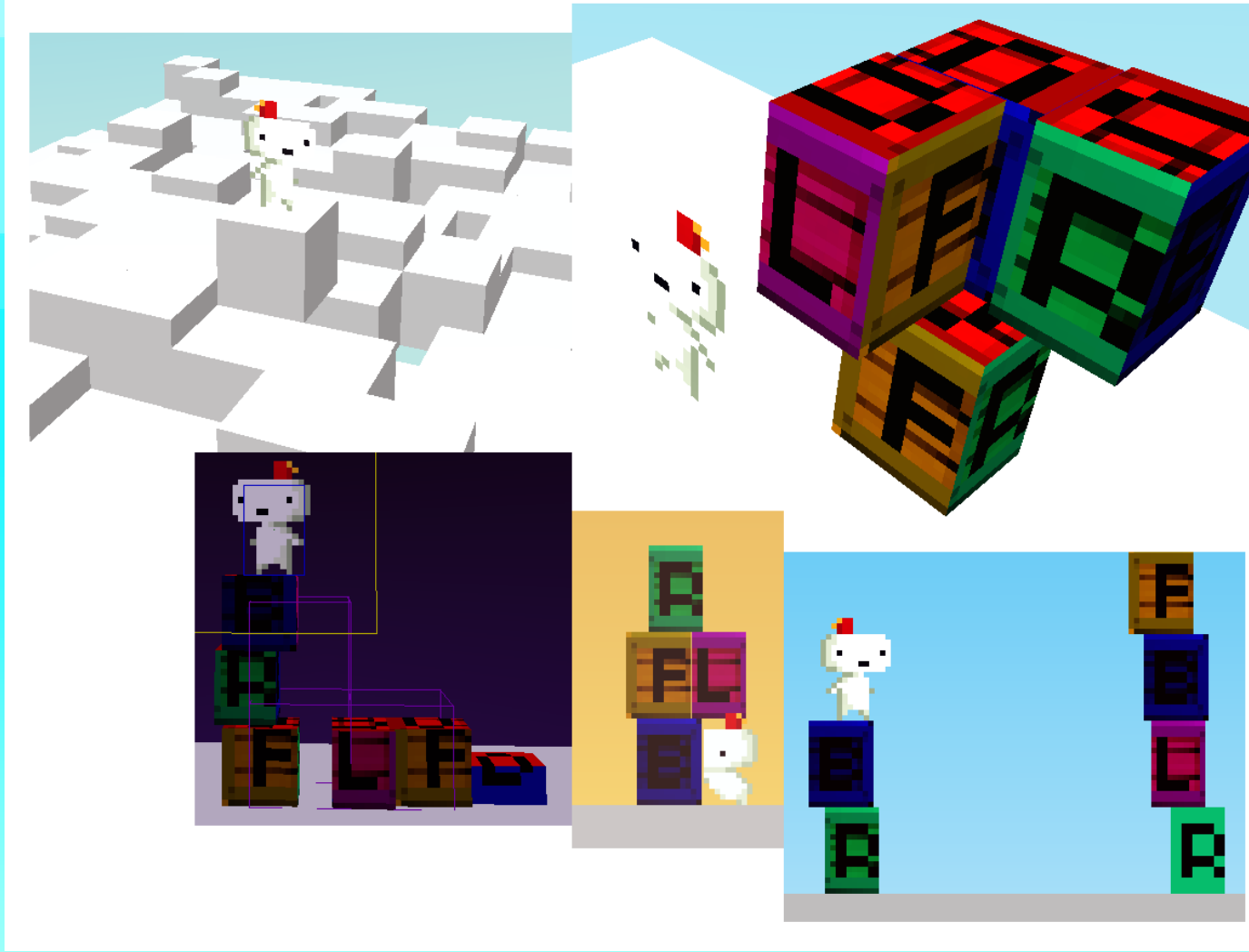
Collision Lookup

- Level is a large structure of triles indexable by (x, y, z)
  - Actually a Dictionary<Int3, Trile>
- You collide with what you see
  - Get closest (x, y) or (z, y) row of triles
  - Traverse to nearest collideable trile and use its properties
- Gets tricky with offset triles
  - Look up 4 closest neighbour rows and test



- After that, normal AABB to line collision resolution

Offset/Non-Unit Tiles



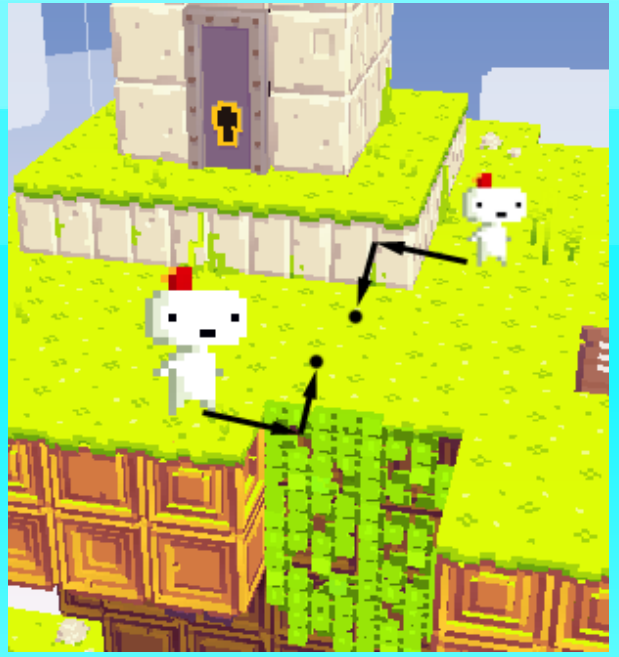
How Gomez Moves Around

- Movement is in 2D according to the current viewpoint.

Depth correction

- Gomez should stay visible at all times
- Gomez should never walk in mid-air

Otherwise, don't change Gomez's depth arbitrarily



- When the camera rotates, movement (and time) is suspended.

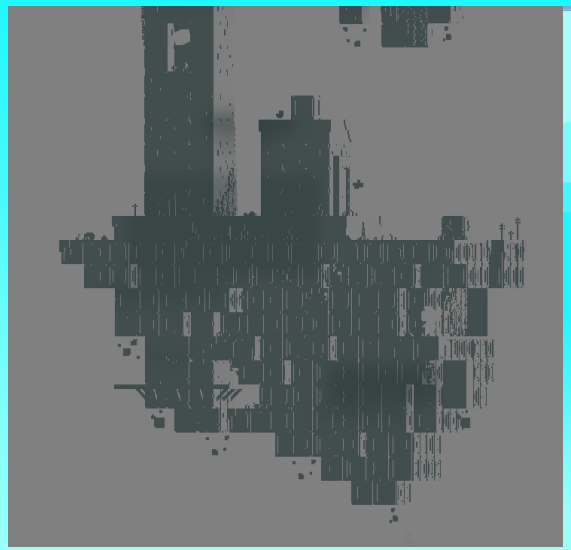
Background Mode

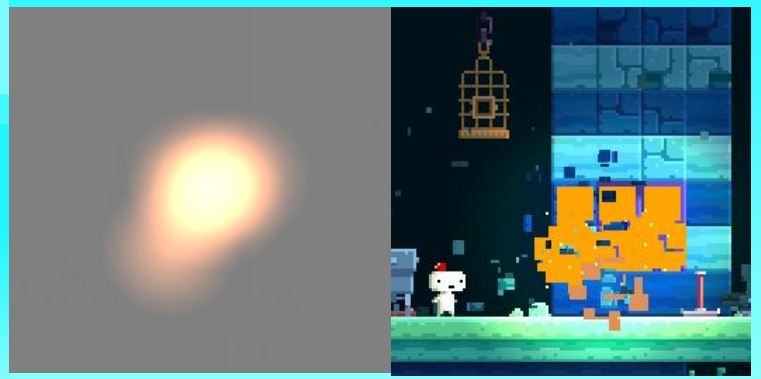
- If Gomez is behind geometry after a rotation, don't panic!

  - Silhouette rendering
  - Low-pass music
  - Limited actions

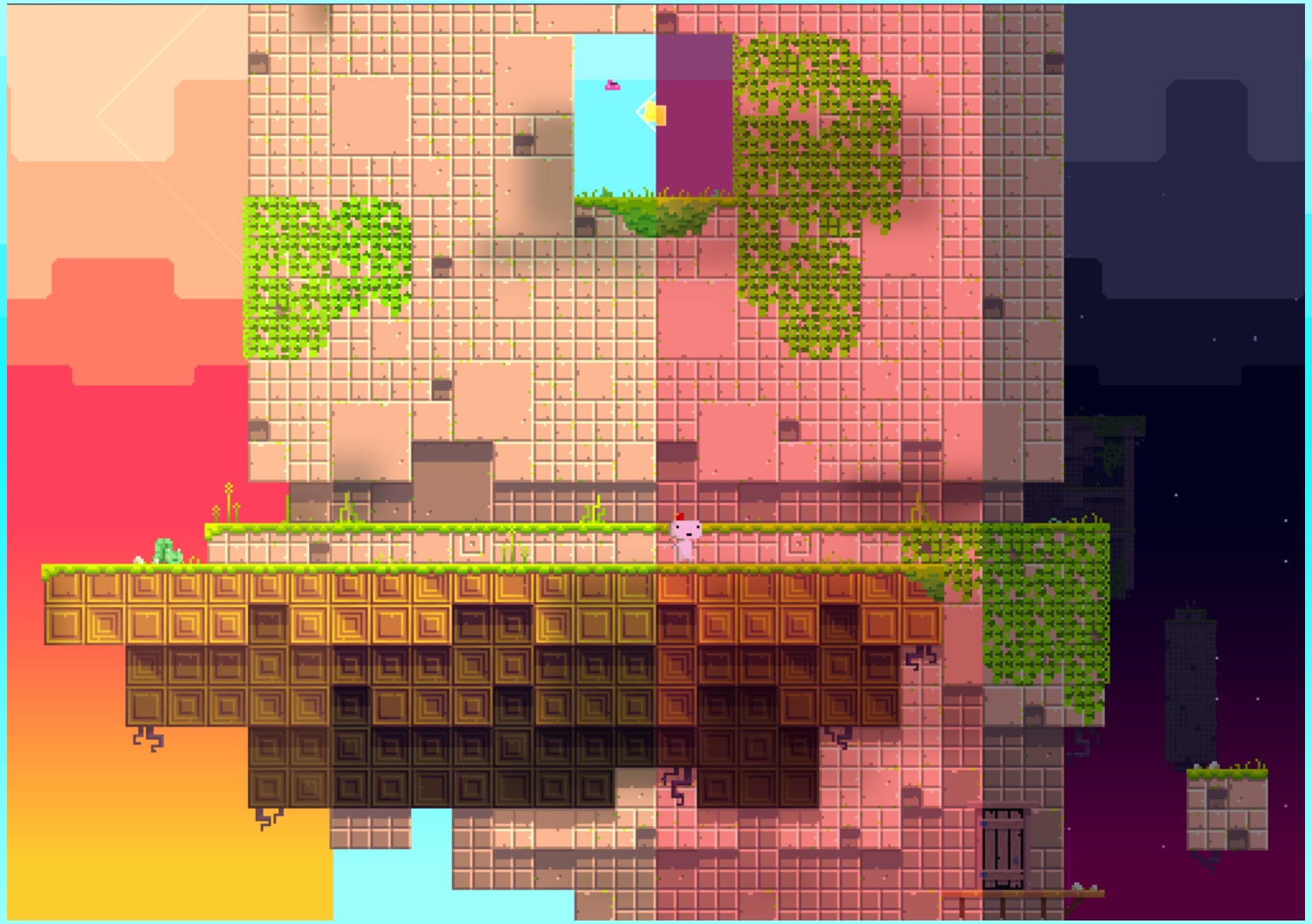
Lighting

- Per-face direct/diffuse light
- Ambient light based on sky background color
  - Also tints transparent cloud layers
- Shadows and additional lights added in screen-space
- All done in a lighting pre-pass (render-to-texture)
  - Blended in Modulate2X mode so that it can light up and shadow





Time of Day Lighting



Dynamic World Interactions

- FEZ @ IGF'08 was 99% static

- Now Gomez can :
  - Grab/push/pull objects
  - Rotate parts of the world independently
  - Make blocks crumble under his weight
  - Grab ledges all around platforms
  - Interact with one-off puzzle objects
  - Swim in water & drown in toxic liquids
  - AND MUCH MUCH MORE!

- 56 different action classes control his behaviour

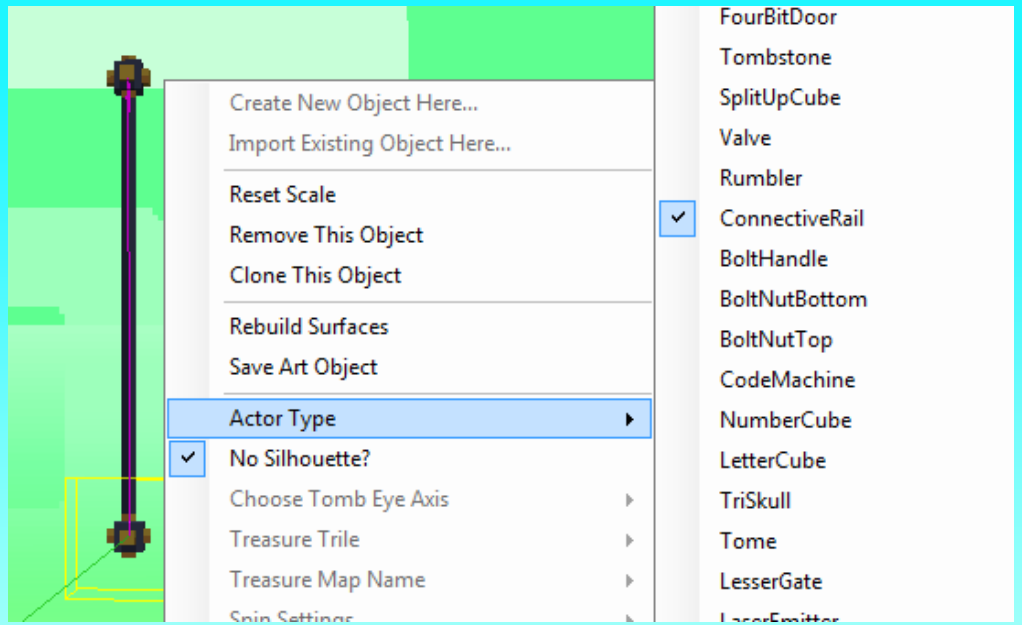
Action "Objects For States"

- All player action classes derive from a base abstract class
- Not all mutually exclusive
  - "Fall" is an action that is evaluated as long as there's gravity
- Tied to sprite animations, player controllability, etc.

```
protected virtual void TestConditions()
{
}

protected virtual void Begin()
{
}

protected virtual void End()
{
}

protected virtual bool Act(TimeSpan elapsed)
{
    return false;
}
```

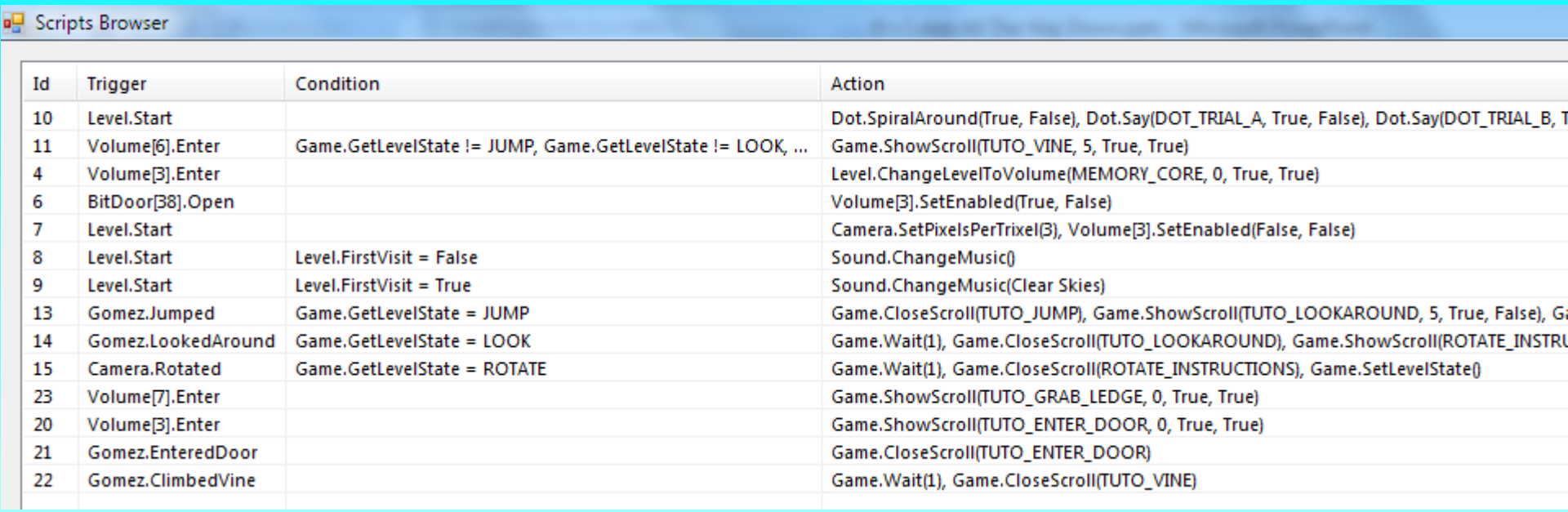
Dynamic World Entities = Actors

- Spinning blocks, interactive structures, etc.
- Hardcoded behaviours with flags or parameters set in the editor
- Tradeoff for not having proper scripting support



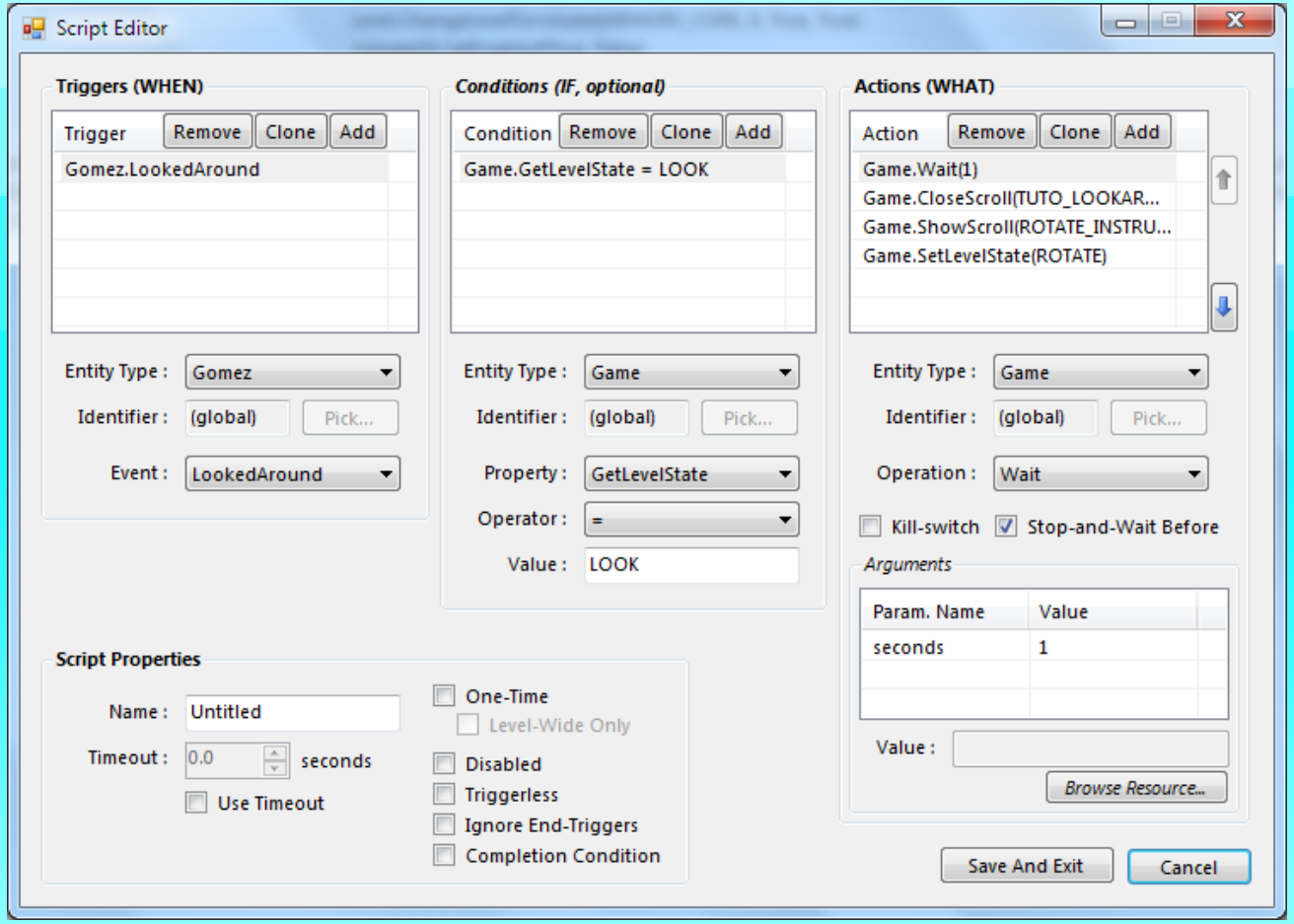
Scripting System

- Designer-friendly UI (no code!)
- Event-based (no continuous evaluation)
- Extensible and backwards-compatible



Final Script Editor UI

...is scary.



Serialized Scripts

- Scripts are objects, serialized as text inside the level file
- Tweakable by hand
  - If you are careful with braces
- Probably should've written a better formatter like :

```
script key=0 {
  code "Level.Start =>
    Camera.SetPixelsPerTrixel(2)"
}
```

```
script key=0 {
  name "Untitled"
  triggers {
    trigger {
      event "Start"
      object {
        type "Level"
      }
    }
  }
  actions {
    action {
      operation "SetPixelsPerTrixel"
      arguments "2"
      object {
        type "Camera"
      }
    }
  }
}
```

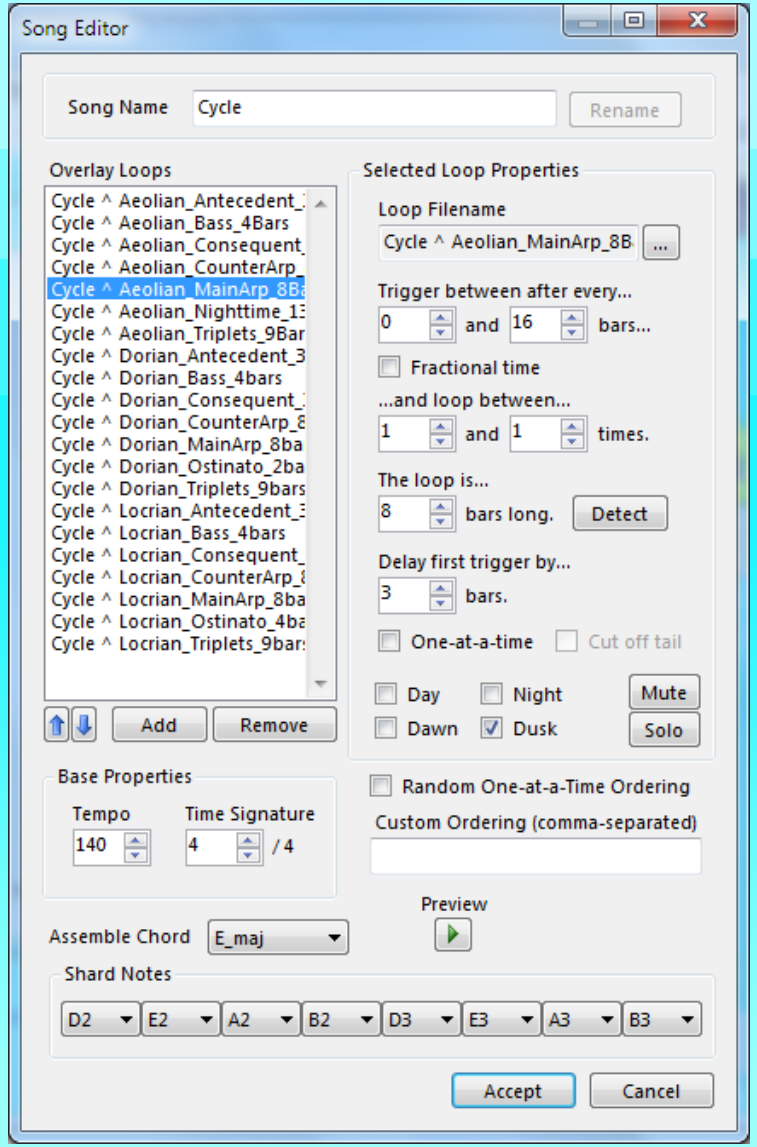
Scripting Interfaces

- Reflected by the editor to fill dropdowns

```
[Entity(Static = true)]
public interface ILevelService : IScriptingBase
{
        // Events
        [Description("When the level starts")]
        event Action Start;
        void OnStart(); // Called by the engine to trigger the event

        // Properties
        bool FirstVisit { get; }

        // Operations
        [Description("Smoothly changes to a new water height")]
        LongRunningAction SetWaterHeight(float height);
}
```

Music System

- Written on-top of XACT
- Allows dynamic track-based songs
- Works with time of day
- Horribly intricate and complicated, as requested by Disasterpeace!
  - ...but allows for kickass songmaking

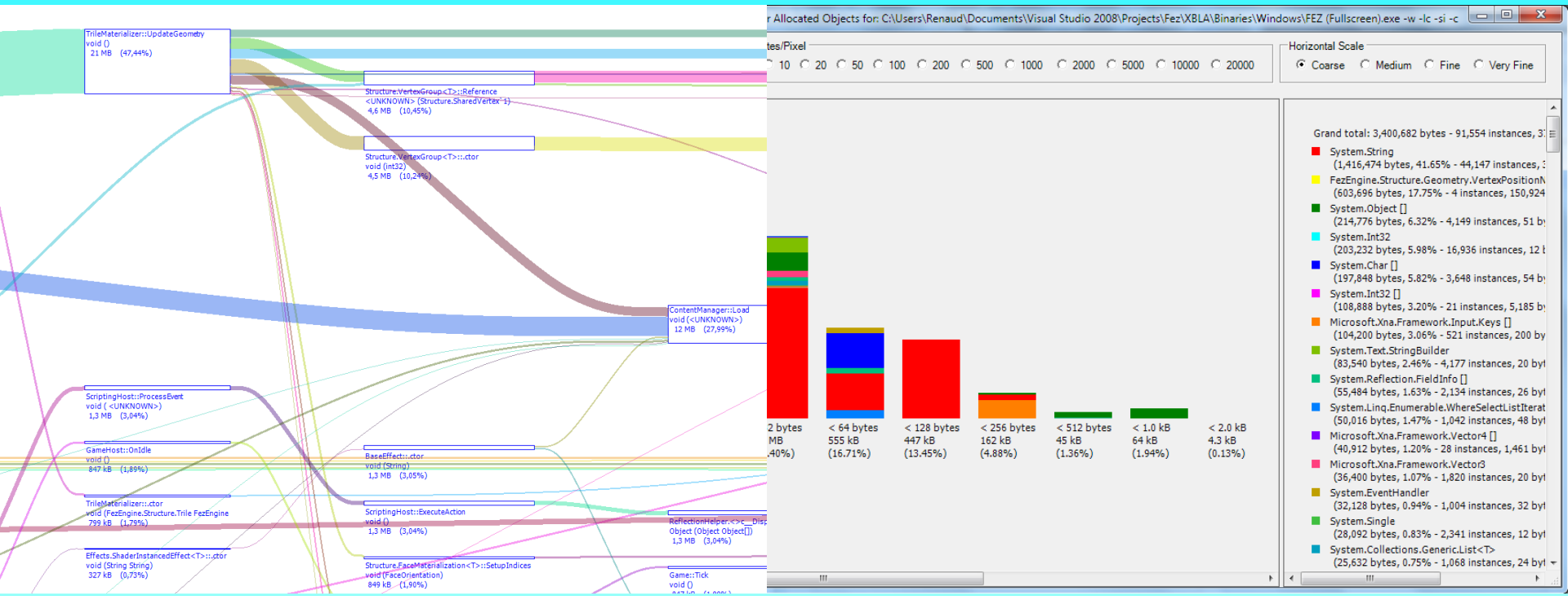


Xbox 360 Optimizations

- XNA on the Xbox 360 = .NET Compact Framework
  - Garbage collection every 1Mb allocated, unpredictable, blocking
  - Draw calls are expensive : batching is essential, instantiate ALL THE THINGS
  - Defer as many draw calculations as possible to vertex shaders instead of CPU
  - Otherwise, multithread; 5 (slow) cores at your disposal
- Lots of content on the Web about this
  - ...especially since XBLIG games have the same issues
  - Rules of thumb : avoid LINQ, dynamic allocations

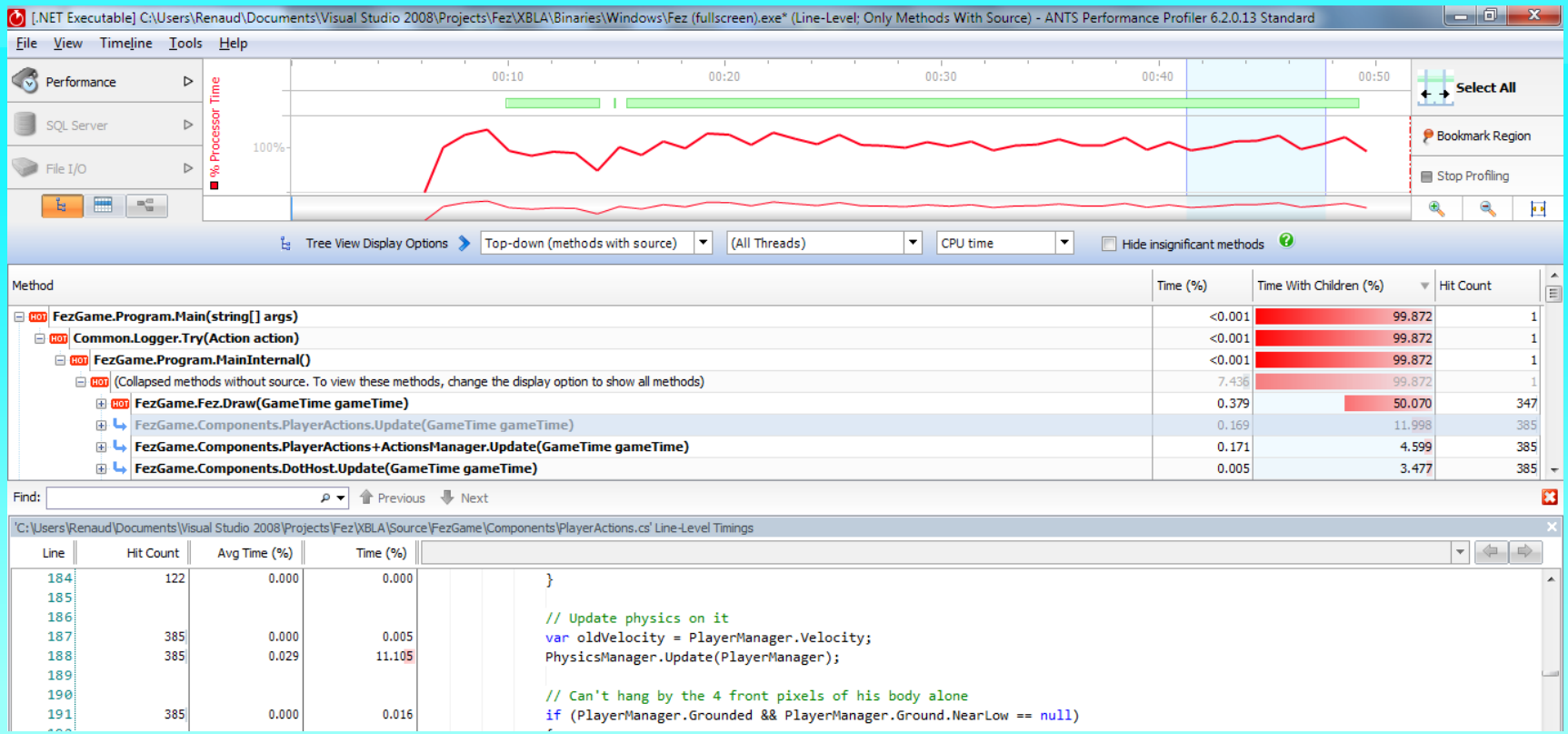
Tools : CLR Profiler

- Excellent, free memory profiler
- Allocations graphs, memory usage by object
- Good for identifying real-time garbage & why load-times stall



Tools : CPU Profiler

- Any one you want (AQTime, dotTrace…)
  - I like ANTS Performance Profiler a lot
- Absolutely essential for finding bottlenecks

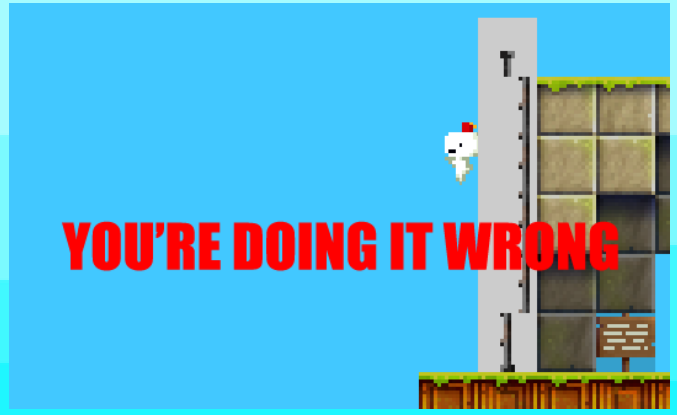


Tools : xbWatson, PIX

- Come with the Xbox Development Kit
- xbWatson
  - Make your own measurements and output to debug
- PIX
  - Frame-by-frame teardown of what's costly
  - Excellent for inspecting the draw calls & shaders
- CPU Performance Profiler in the XDK ultimately not that useful
  - Made for native code, not .NET games

WHAT TOOK YOU SO LONG?

- Game started in April 2007
- Still in QA in November 2011



- What the hell is going on??

Timeline

- **2007** : Groundwork, engine, editor, “level one”
  - ...for the IGF’08 prototype, eventually scrapped (built in 100 days)
- **2008** : Celebration, setting up Polytron, art re-draw, engine work while finishing studies, more scrapped levels
- **2009** : Full-time work begins, world design, art & more engine work
- **2010** : Design dialogue with Microsoft, internal drama, DOT and world map come in, BMcC & Disasterpeace come in (October)
- **2011** : Puzzle & cutscene work, more levels, QA, optimization and polishing

That's all, folks!

- Thanks for coming!
- Questions?