GameMaker Studio 2

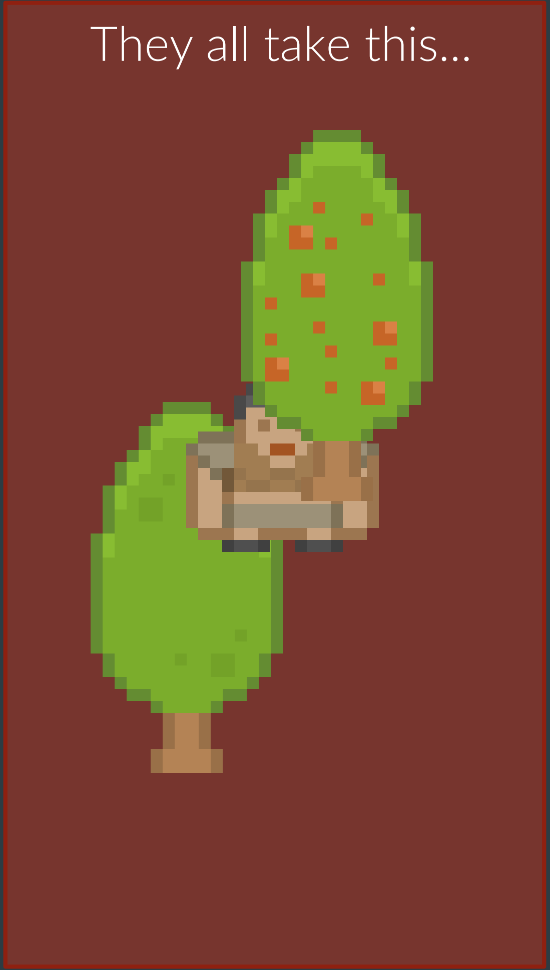
2.5D Object-Depth Systems

MirthCastle

Common GameMaker Depth Systems

- Grid-Sort (Friendly Cosmonaut)
- Binary-Lists (Ariak)
- Grid-Lists (YoYo Games)
  - YoYo Dungeon Tutorial
- Grid-Layers (MirthCastle)
  - This Tutorial

They all take this…



GameMaker Studio
Default Sorting

On the left you can see where the depth sorting is clearly wrong for a 2.5D game… unless your character can grow a tree from his chest!

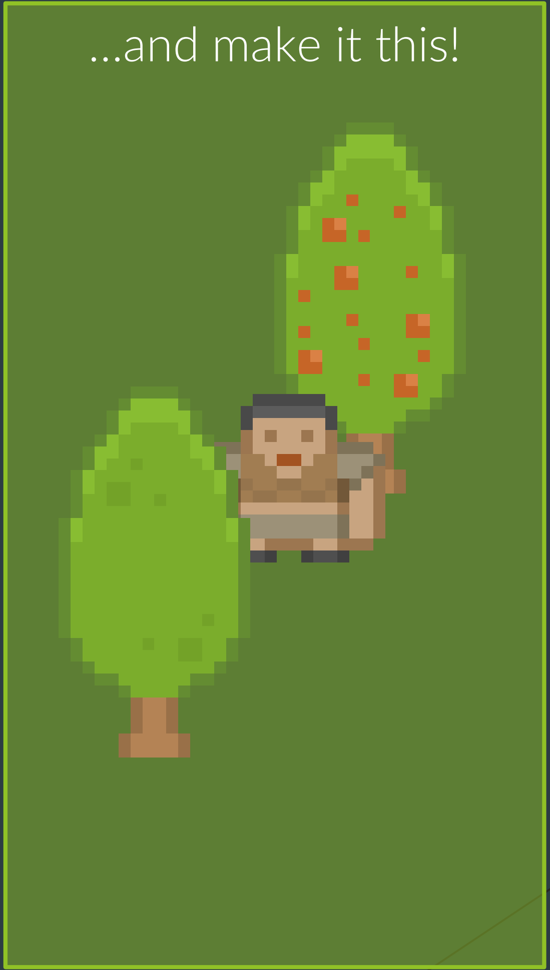
MirthCastle

Common GameMaker Depth Systems

- Grid-Sort (Friendly Cosmonaut)
- Binary-Lists (Ariak)
- Grid-Lists (YoYo Games)
  - YoYo Dungeon Tutorial
- Grid-Layers (MirthCastle)
  - This Tutorial

That is much more like it! Though I'm calling dibs on the game where the guy has the power to grow trees from his chest!

...and make it this!



Sorting via Depth System

MirthCastle

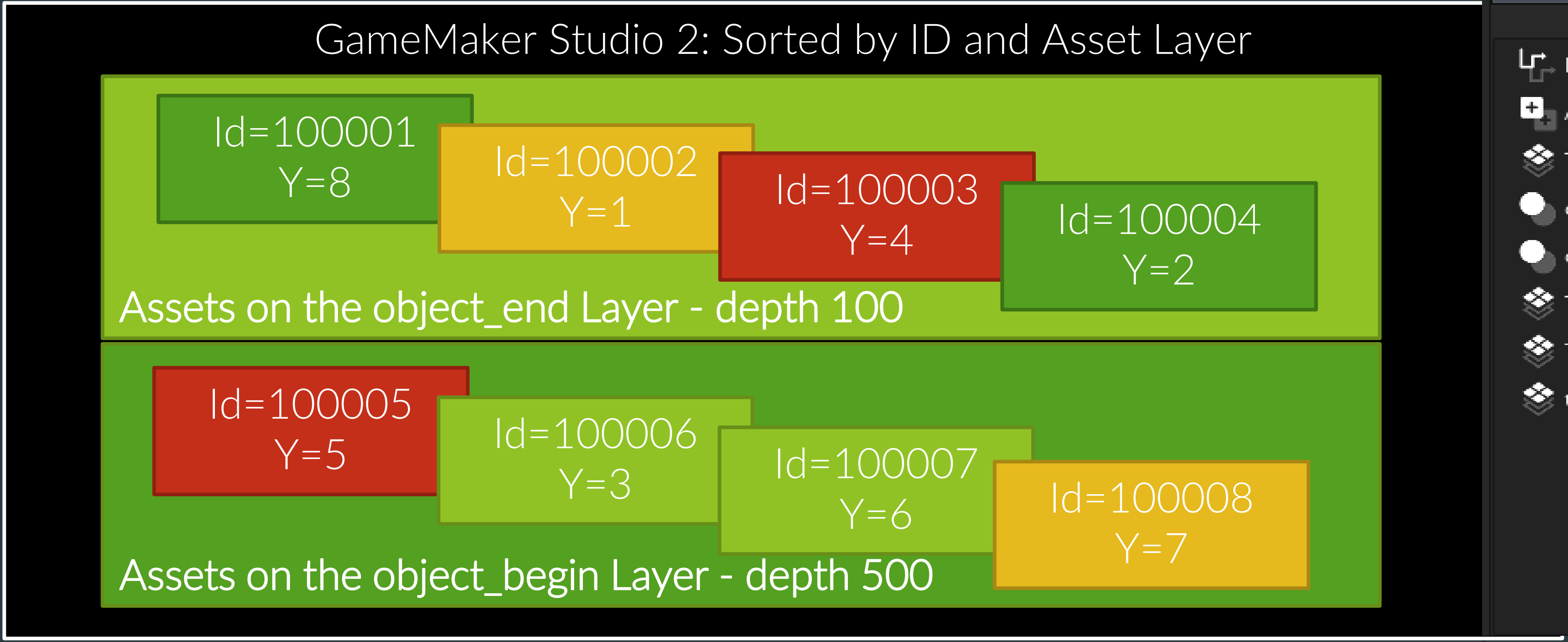
WARNING: Technical Mumbo-Jumbo Follows!

Q: Why didn’t the chicken cross the road?...

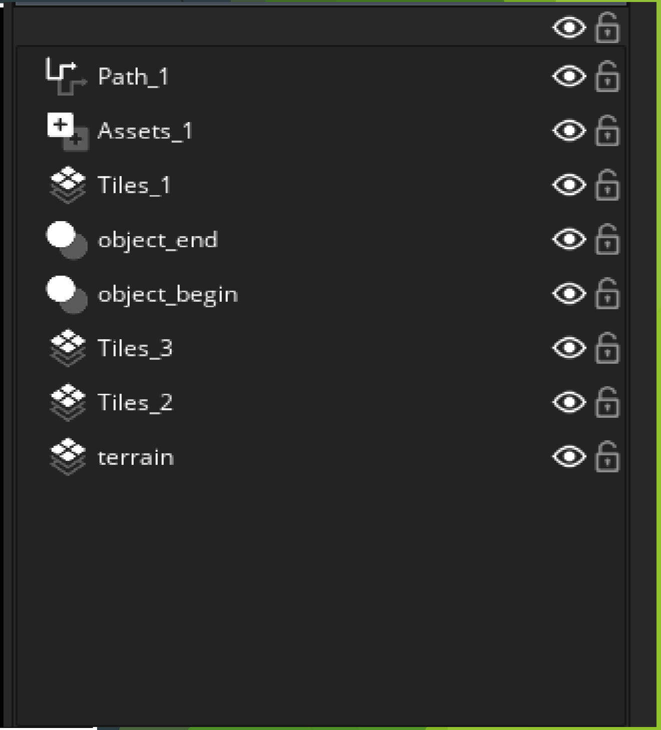
How does GameMaker do it?

Why do we need this?

GameMaker Studio 2 (GMS2) does not have a **specific** 2.5D depth system. In the end GMS2 processes the Draw-Events of every instance by "order" of their **Layer-depth** highest to lowest, and then their **instance ID #** lowest to highest! The **ID** #s of instances are set when they are created, and we have almost **no** control over this.



GameMaker Studio 2: Sorted by ID and Asset Layer
Id=100001 Y=8
Id=100002 Y=1
Id=100003 Y=4
Id=100004 Y=2
Assets on the object_end Layer - depth 100
Id=100005 Y=5
Id=100006 Y=3
Id=100007 Y=6
Id=100008 Y=7
Assets on the object_begin Layer - depth 500



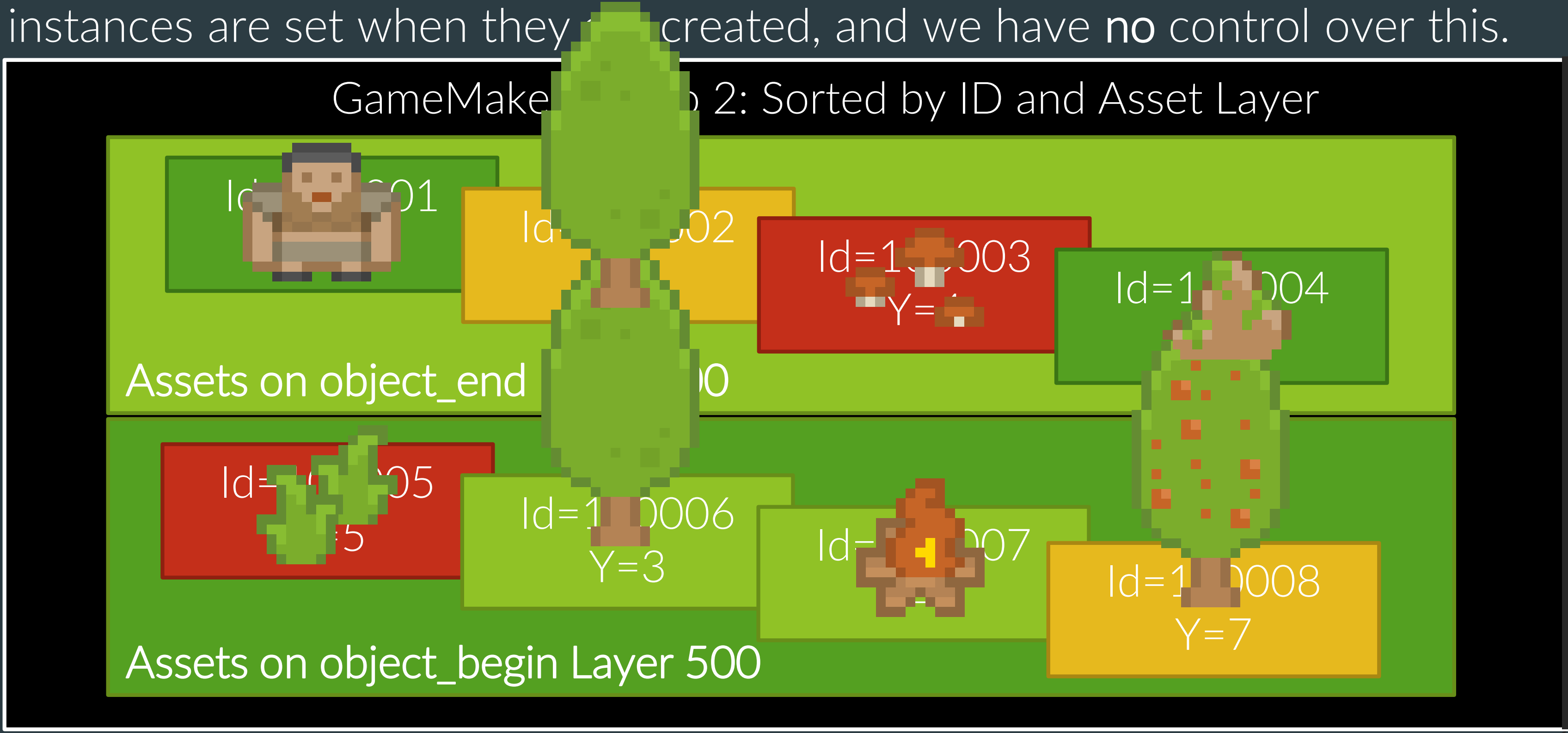
The assets above are drawn by **ID** #s low to high so the highest **ID** #s are drawn **LAST**. It is the opposite for Layers, however. **Higher** Layer-depths are drawn FIRST. So above all the assets on Layer 100 **overlap** Layer 500, even though the instance **ID** #s are higher.

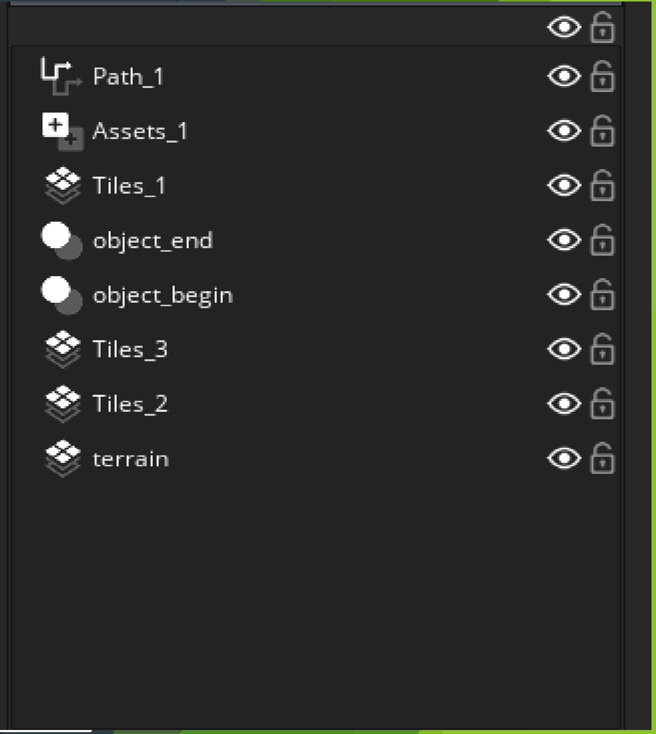
MirthCastle

How does GameMaker do it?

Why do we need this?

GameMaker Studio 2 (GMS2) does not have a specific 2.5D depth system. In the end GMS2 processes the Draw-Events of every instance by "order" of their **Layer-depth** highest to lowest, and then their **instance ID #** lowest to highest! The **ID** #s of instances are set when they are created, and we have **no** control over this.


GameMaker Studio 2: Sorted by ID and Asset Layer
Id=100001
Id=100002
Id=100003
Y=
Id=100004
Assets on object_end Layer 400
Id=100005
Y=5
Id=100006
Y=3
Id=100007
Id=100008
Y=7
Assets on object_begin Layer 500



The result above shows that the GMS2 built-in sorting methods could still end up with some mistakes. Computers are dumb! This is where **we** the developers need to step in and tell the computer what to do.

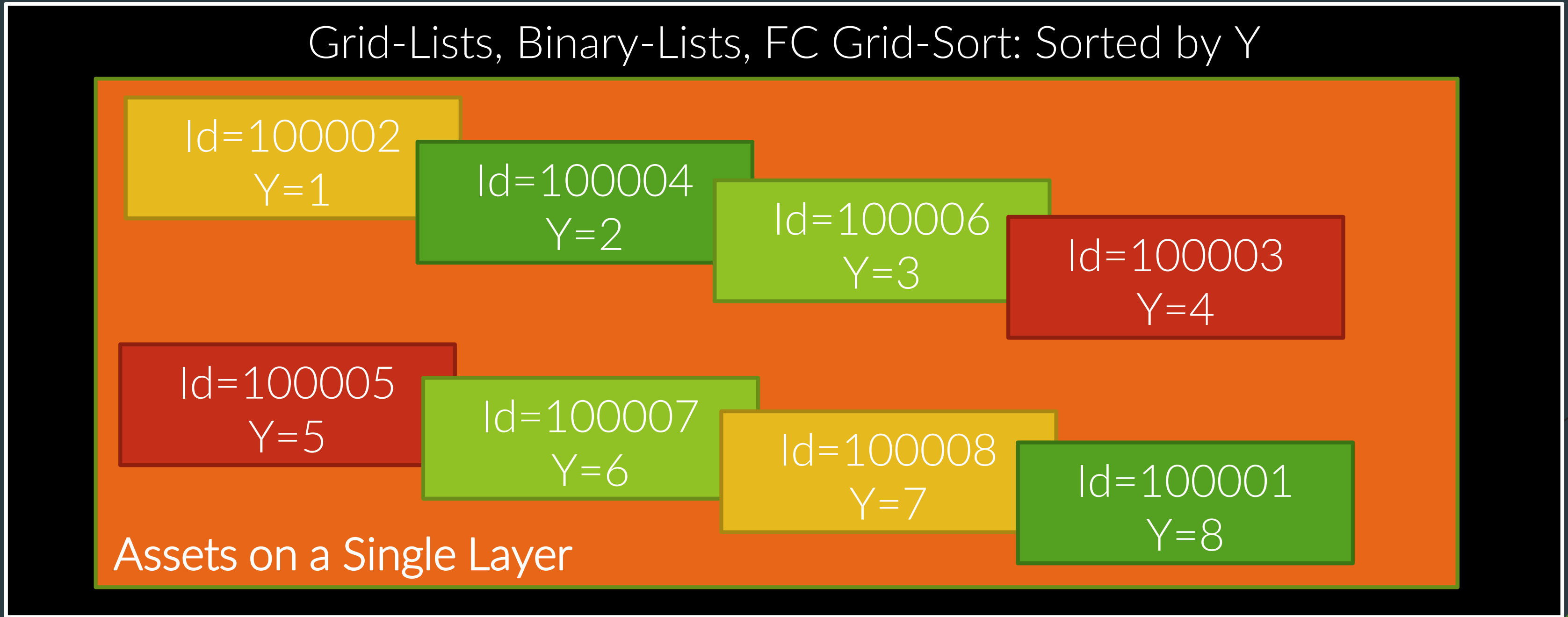
MirthCastle

How do the others work?

Binary-Lists (Ariak) and FC Grid-Sort (Friendly Cosmonaut) method is to collect all of the instances into a **ds_list**, **ds_grid**, and or an **array**, and then use the Y value of the instances to sort the "list" into the draw-order we want. The drawback is the longer the list, the **longer** this sorting step takes.

Grid-Lists (YoYo Games) method creates a **ds_grid**\array of cells the size of the room vertically, and creates a **ds_list** in each. The instances use their Y value to add themselves to the correct **ds_list** in the **ds_grid.** It then loops through the **ds_grid**, looping through each **ds_list** in turn for the draw-order. The drawback is it takes a **long** time to loop through the entire grid-of-lists. The **bonus** is we skip the sort step.

All three systems also suffer a **bottleneck** and drawback within the Draw-Event. To work they all use a controller-object that **manually** loops **each** and **every** item in their lists, or grid-of-lists, and then forcefully draws that instance to screen itself by overriding the others Draw-Event, one-at-a-time, on a single layer. This puts a **heavy** load on GMS2, **and** limits our ability to draw effects like shadows, silhouettes, or blends.



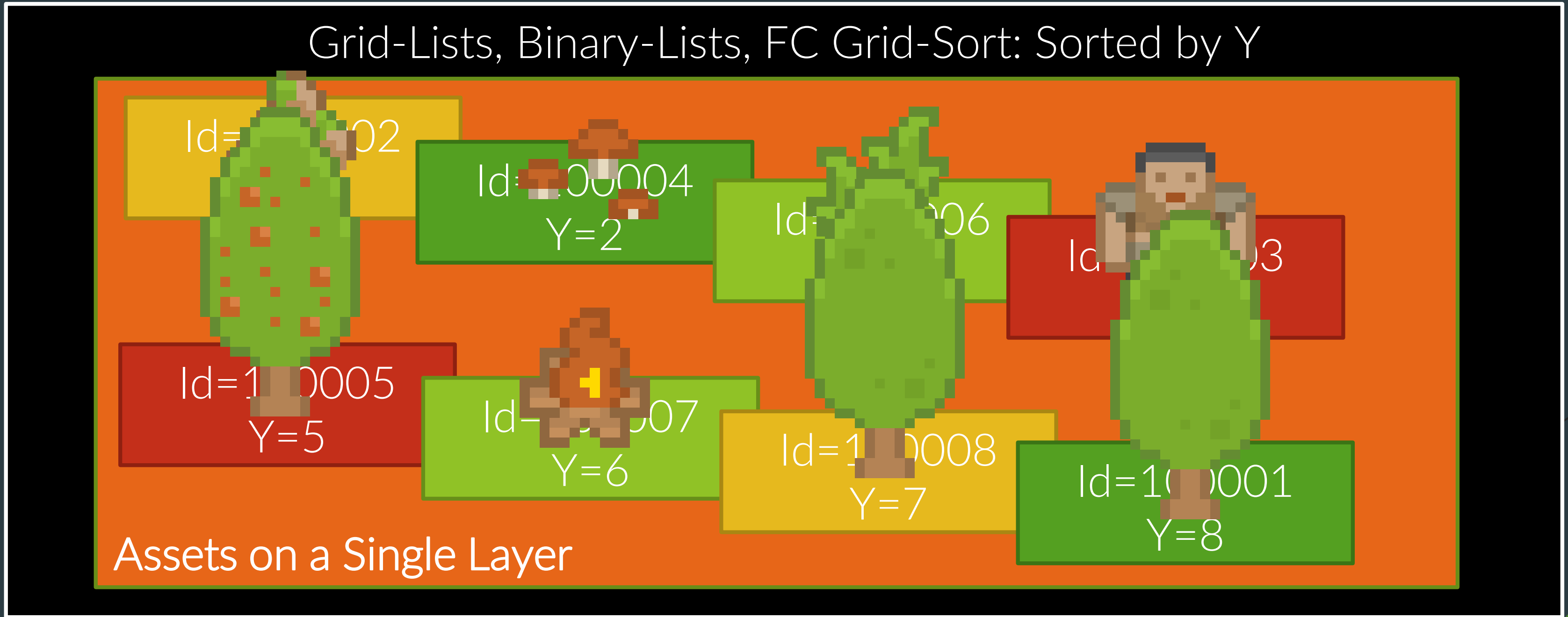
MirthCastle

How do the others work?

Binary-Lists (Ariak) and FC Grid-Sort (Friendly Cosmonaut) method is to collect all of the instances into a ds_list, ds_grid, and or an array, and then use the Y value of the instances to sort the "list" into the draw-order we want. The drawback is the longer the list, the longer this sorting step takes.

Grid-Lists (YoYo Games) method creates a ds_grid\array of cells the size of the room vertically, and creates a ds_list in each. The instances use their Y value to add themselves to the correct ds_list in the ds_grid. It then loops through the ds_grid, looping through each ds_list in turn for the draw-order. The drawback is it takes a long time to loop through the entire grid-of-lists. The bonus is we skip the sort step.

All three systems also suffer a bottleneck and drawback within the Draw-Event. To work they all use a controller-object that manually loops each and every item in their lists, or grid-of-lists, and then forcefully draws that instance to screen itself by overriding the others Draw-Event, one-at-a-time, on a single layer. This puts a heavy load on GMS2, and limits our ability to draw effects like shadows, silhouettes, or blends



A heavy price, but they are sorted correctly!

MirthCastle

WARNING: INCOMING MS Excel Abuse!

A: ...because he was such a chicken!!! :P

MirthCastle

Introducing Grid-Layers – What's New?

ds_grid-cells – hold LayerIDs	Code-Created Layers
Grid[0, 0] = y0 = LayerID layer_depth = 500	Instances, Sprites, Assets Batch-Drawn at the **Layer Depth** – **Skipped** if empty
Grid[0, 1] = y1 = LayerID layer_depth = 475	Instances, Sprites, Assets Batch-Drawn at the **Layer Depth** - **Skipped** if empty
Grid[0, 2] = y2 = LayerID layer_depth = 450	Instances, Sprites, Assets Batch-Drawn at the **Layer Depth** - **Skipped** if empty
Grid[0, 3] = y3 = LayerID layer_depth = 425	Instances, Sprites, Assets Batch-Drawn at the **Layer Depth** - **Skipped** if empty
Grid[0, 4] = y4 = LayerID layer_depth = 400	Instances, Sprites, Assets Batch-Drawn at the **Layer Depth** - **Skipped** if empty
Grid[0, 5] = y5 = LayerID layer_depth = 375	Instances, Sprites, Assets Batch-Drawn at the **Layer Depth** - **Skipped** if empty
Grid[0, ++] = y++ = LayerID layer_depth = XXXX	Instances, Sprites, Assets Batch-Drawn at the **Layer Depth** - **Skipped** if empty

What's New?

Faster than **every** other sorting system tested.

Assets are sorted between Layers, not depths, grids, or lists.

"Static" objects are **only** sorted **ONCE,** and are deactivated outside View**.**

"Active" objects are **only** sorted on Y axis changes.

Draw_Events are **NOT** overridden by a single controller-object.

Layers are managed via code at any time.

MirthCastle

Introducing Grid-Layers – What's Different?

ds_grid-cells – hold LayerIDs	Code-Created Layers
Grid[0, 0] = y0 = LayerID	
layer_depth = 500	
Grid[0, 1] = y1 = LayerID	
layer_depth = 475	
Grid[0, 2] = y2 = LayerID	
layer_depth = 450	
Grid[0, 3] = y3 = LayerID	
layer_depth = 425	
Grid[0, 4] = y4 = LayerID	
layer_depth = 400	
Grid[0, 5] = y5 = LayerID	
layer_depth = 375	
Grid[0, ++] = y++ = LayerID	
layer_depth = XXXX	

What's Different?

Grid-Layers is *similar* to Grid-Lists in that it uses a **ds_grid** to reference every Y position in the room. Instances use their Y position to match up with a **ds_grid-cell** to reference where to go.

However, instead of a **ds_list** reference in each cell, Grid-Layers creates an **actual Layer** reference for the instances to be sorted to.

MirthCastle

Introducing Grid-Layers – Why Layers?

ds_grid-cells – hold LayerIDs	Code-Created Layers
Grid[0, 0] = y0 = LayerID	
layer_depth = 500	
Grid[0, 1] = y1 = LayerID	
layer_depth = 475	
Grid[0, 2] = y2 = LayerID	
layer_depth = 450	
Grid[0, 3] = y3 = LayerID	
layer_depth = 425	
Grid[0, 4] = y4 = LayerID	
layer_depth = 400	
Grid[0, 5] = y5 = LayerID	
layer_depth = 375	
Grid[0, ++] = y++ = LayerID	
layer_depth = XXXX	

Why Layers?

GMS2 Layers are automatically drawn by order of depth, high to low, so no sorting of Y values from a ds_list or ds_grid.

Layers in GMS2 batch-process all the activated assets assigned to them for drawing all-at-once, so no ds_list to loop through at each Y either.

Empty Layers are simply Skipped without further checks.

MirthCastle

Introducing Grid-Layers – Recap?

ds_grid-cells – hold LayerIDs	Code-Created Layers
Grid[0, 0] = y0 = LayerID	
layer_depth = 500	
Grid[0, 1] = y1 = LayerID	
layer_depth = 475	
Grid[0, 2] = y2 = LayerID	
layer_depth = 450	
Grid[0, 3] = y3 = LayerID	
layer_depth = 425	
Grid[0, 4] = y4 = LayerID	
layer_depth = 400	
Grid[0, 5] = y5 = LayerID	
layer_depth = 375	
Grid[0, ++] = y++ = LayerID	
layer_depth = XXXX	

Recap?

With Grid-Layers: Every asset, on every Layer, is automatically batch-processed by order of the Layer-depth they are assigned, high to low… automatically! :P

No ds_grid loops, No ds_list loops, and No Draw-Event overrides.

Grid-Layers works WITH GMS2 built-in tools. We simply help the instance find the Layer it needs, and GMS2 does the rest.

MirthCastle

WARNING: WALL OF TEXT!! @.@

MirthCastle

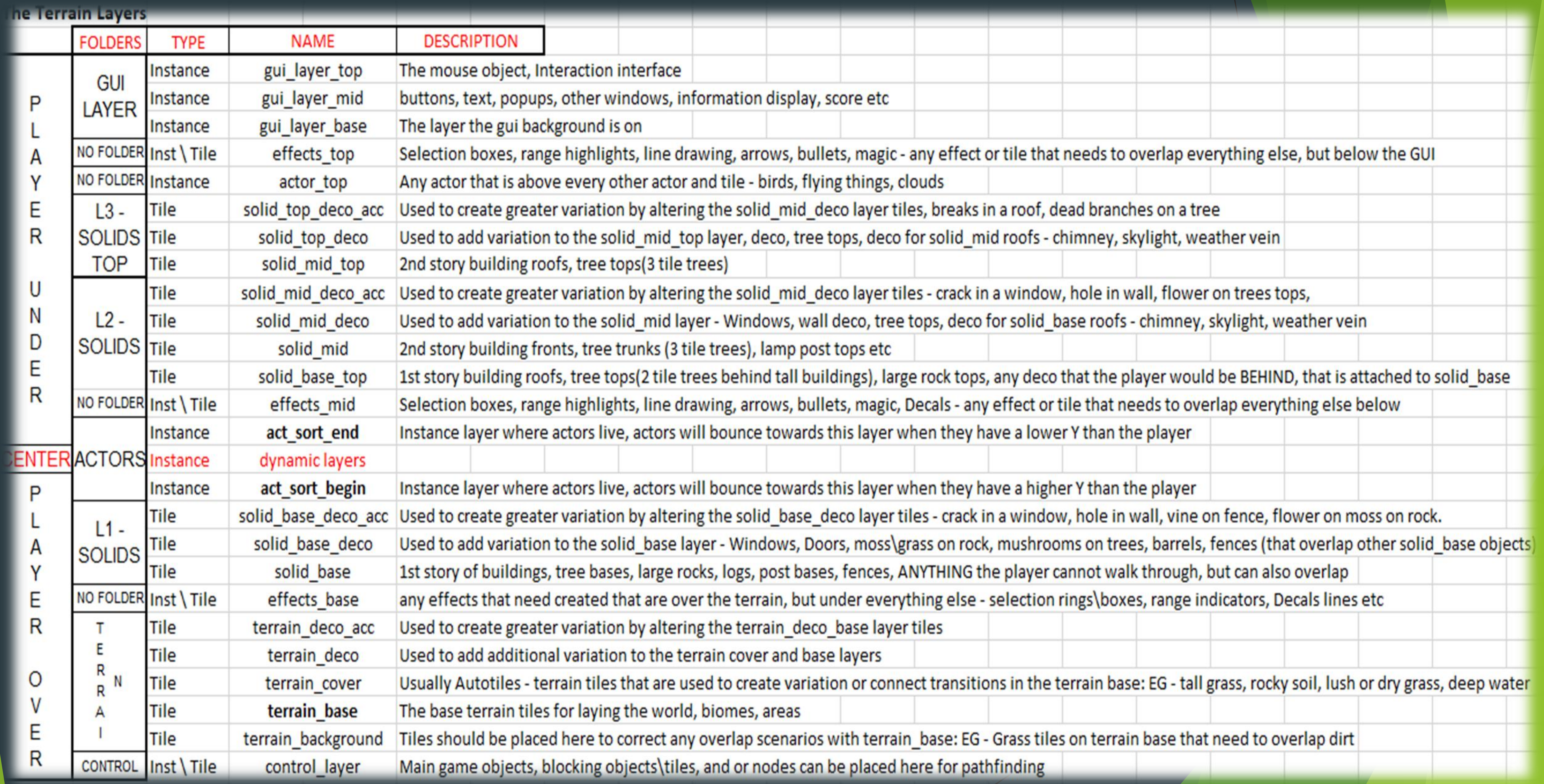
Grid-Layers – Major Bullet-Points

Grid-Layers takes legit advantage of the highly-optimized Layer system in GMS2. Instances move themselves between **Layers**, not depths, using a ds_grid as an **easy** reference to the Layer they need moved onto. This is NOT depth = -y.

- **Faster** than Binary Lists and FC Grid Sort even at low instance counts, AND **faster** than Grid-Lists at >500 instances with far less overhead.
- Grid-Layers gives back FULL CONTROL over instances Draw_Events. It does **not** use a controller-object that overrides instances, so **no bottleneck!** This makes effects like shadows and blends easy again!
- Just one **ds_grid\array** for easy Layer reference that doesn't need looped through.
- "Static" objects **only** need sorted to a Layer **ONCE**, while "Active" objects **only** need sorted to a new Layer when they **move** on the Y axis, and\or are within the View. No other sorting required.
- Out of View "Active" instances can be deactivated, AI slowed down (less searching for player?), ignored (no sorting), or left alone (sorting gets turned off if they are **outside** the room **vertically**).
- Out of View "Static" instances get deactivated to **speed up** the system even further.
- **Code-created Layers** are also controllable via code and **not** limited to just instances. Sprites, and other Assets can all be drawn and sorted on them too. The Layers can also be disabled to allow for multiple Views **without** double-drawing everything.
- Rounding instance Y values to a power of 2 dramatically **reduces memory** and needed Layers.

MirthCastle

Is That Your Layers??



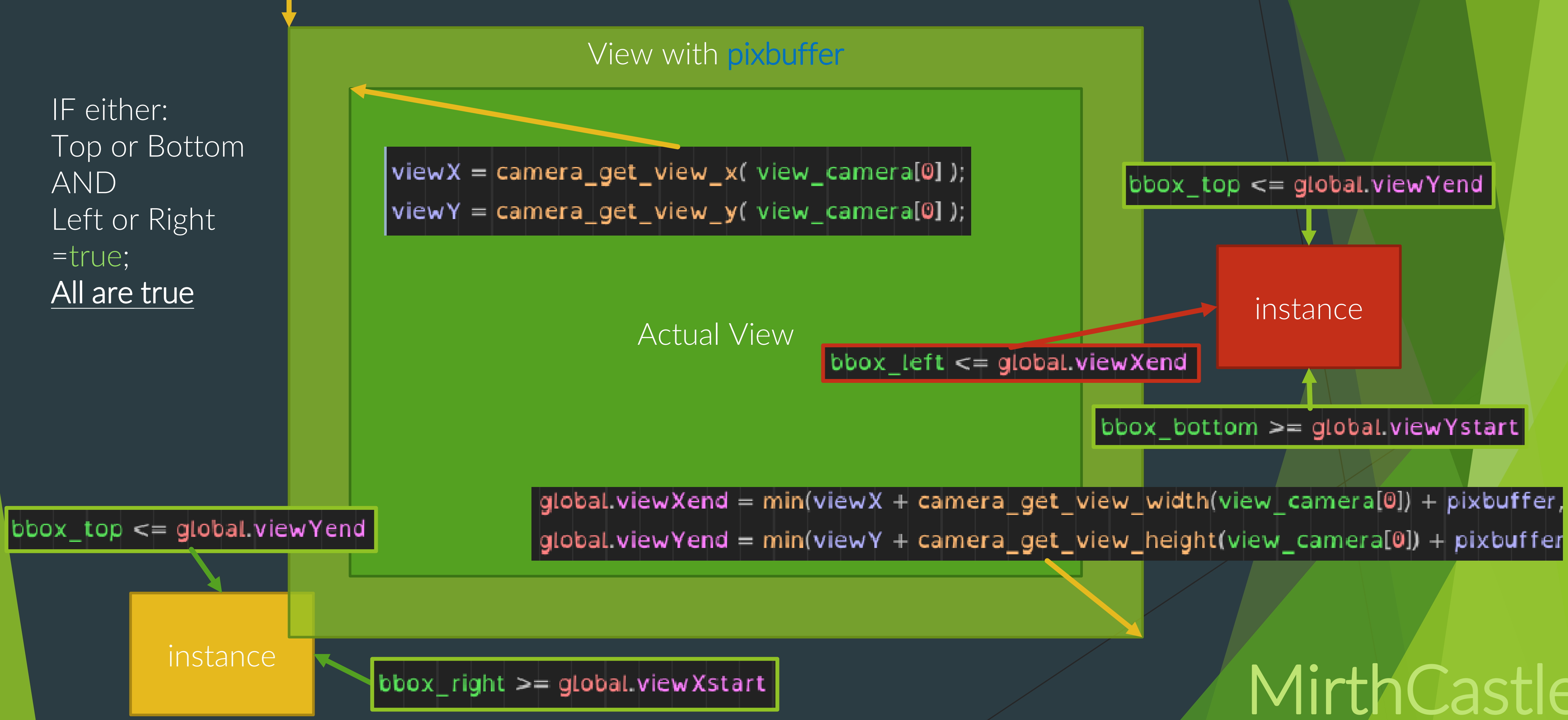
he Terrain Layers

	FOLDERS	TYPE	NAME	DESCRIPTION
PLAYER UNDER	GUI LAYER	Instance	gui_layer_top	The mouse object, Interaction interface
		Instance	gui_layer_mid	buttons, text, popups, other windows, information display, score etc
		Instance	gui_layer_base	The layer the gui background is on
	NO FOLDER	Inst \ Tile	effects_top	Selection boxes, range highlights, line drawing, arrows, bullets, magic - any effect or tile that needs to overlap everything else, but below the GUI
	NO FOLDER	Instance	actor_top	Any actor that is above every other actor and tile - birds, flying things, clouds
	L3 - SOLIDS TOP	Tile	solid_top_deco_acc	Used to create greater variation by altering the solid_mid_deco layer tiles, breaks in a roof, dead branches on a tree
		Tile	solid_top_deco	Used to add variation to the solid_mid_top layer, deco, tree tops, deco for solid_mid roofs - chimney, skylight, weather vein
		Tile	solid_mid_top	2nd story building roofs, tree tops(3 tile trees)
	L2 - SOLIDS	Tile	solid_mid_deco_acc	Used to create greater variation by altering the solid_mid_deco layer tiles - crack in a window, hole in wall, flower on trees tops,
		Tile	solid_mid_deco	Used to add variation to the solid_mid layer - Windows, wall deco, tree tops, deco for solid_base roofs - chimney, skylight, weather vein
		Tile	solid_mid	2nd story building fronts, tree trunks (3 tile trees), lamp post tops etc
		Tile	solid_base_top	1st story building roofs, tree tops(2 tile trees behind tall buildings), large rock tops, any deco that the player would be BEHIND, that is attached to solid_base
	NO FOLDER	Inst \ Tile	effects_mid	Selection boxes, range highlights, line drawing, arrows, bullets, magic, Decals - any effect or tile that needs to overlap everything else below
	ACTORS	Instance	**act_sort_end**	Instance layer where actors live, actors will bounce towards this layer when they have a lower Y than the player
CENTER		Instance	dynamic layers	
		Instance	**act_sort_begin**	Instance layer where actors live, actors will bounce towards this layer when they have a higher Y than the player
PLAYER OVER	L1 - SOLIDS	Tile	solid_base_deco_acc	Used to create greater variation by altering the solid_base_deco layer tiles - crack in a window, hole in wall, vine on fence, flower on moss on rock.
		Tile	solid_base_deco	Used to add variation to the solid_base layer - Windows, Doors, moss\grass on rock, mushrooms on trees, barrels, fences (that overlap other solid_base objects)
		Tile	solid_base	1st story of buildings, tree bases, large rocks, logs, post bases, fences, ANYTHING the player cannot walk through, but can also overlap
	NO FOLDER	Inst \ Tile	effects_base	any effects that need created that are over the terrain, but under everything else - selection rings\boxes, range indicators, Decals lines etc
	TERRAIN	Tile	terrain_deco_acc	Used to create greater variation by altering the terrain_deco_base layer tiles
		Tile	terrain_deco	Used to add additional variation to the terrain cover and base layers
		Tile	terrain_cover	Usually Autotiles - terrain tiles that are used to create variation or connect transitions in the terrain base: EG - tall grass, rocky soil, lush or dry grass, deep water
		Tile	**terrain_base**	The base terrain tiles for laying the world, biomes, areas
		Tile	terrain_background	Tiles should be placed here to correct any overlap scenarios with terrain_base: EG - Grass tiles on terrain base that need to overlap dirt
	CONTROL	Inst \ Tile	control_layer	Main game objects, blocking objects\tiles, and or nodes can be placed here for pathfinding

```
bbox_right >= global.viewXstart && bbox_bottom >= global.viewYstart && bbox_left <= global.viewXend && bbox_top <= global.viewYend
```

```
global.viewXstart = max(viewX - pixbuffer, 0);
global.viewYstart = max(viewY - pixbuffer, 0);
```

IF either:
Top or Bottom
AND
Left or Right
=true;
<u>All are true</u>



```
viewX = camera_get_view_x( view_camera[0] );
viewY = camera_get_view_y( view_camera[0] );
```

```
bbox_top <= global.viewYend
```

```
bbox_left <= global.viewXend
```

```
bbox_bottom >= global.viewYstart
```

```
global.viewXend = min(viewX + camera_get_view_width(view_camera[0]) + pixbuffer,
global.viewYend = min(viewY + camera_get_view_height(view_camera[0]) + pixbuffer
```

```
bbox_top <= global.viewYend
```

```
bbox_right >= global.viewXstart
```

MirthCastle

Closing Thoughts - Tips

- Use tiles in the room_layers to replace objects where you can.
  - This goes for ANY sorting system, even GMS2 default.
- Keep the CELL size as large as you can.
  - Best rule: CELL = speed of slowest instance (min 2)
- Power of 2 for everything practical: Sprites, Maths, Rooms, Backgrounds, etc…
  - Use: 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096
  - This also pleases the GPU gods, especially when width and height match.
- Learn to stay organized with your objects and project
  - Inherit from parents as often as feasible to reduce code
  - Use scripts in place of events or actions that repeat code when able!
  - Alarms are amazing tools! So are Custom User\Other Events!!! Use them!
- ONLY use "Persistence" when you absolutely have to…
  - Keep it to a minimum number of objects if you do, like controller objects, or the player
    - NEVER make a room persistent – unless you plan to write a save file later, and just getting things setup for now, but DON'T rely on it. \`O.O\`
  - Then promise yourself you will learn to write save data to .ini or .txt

MirthCastle