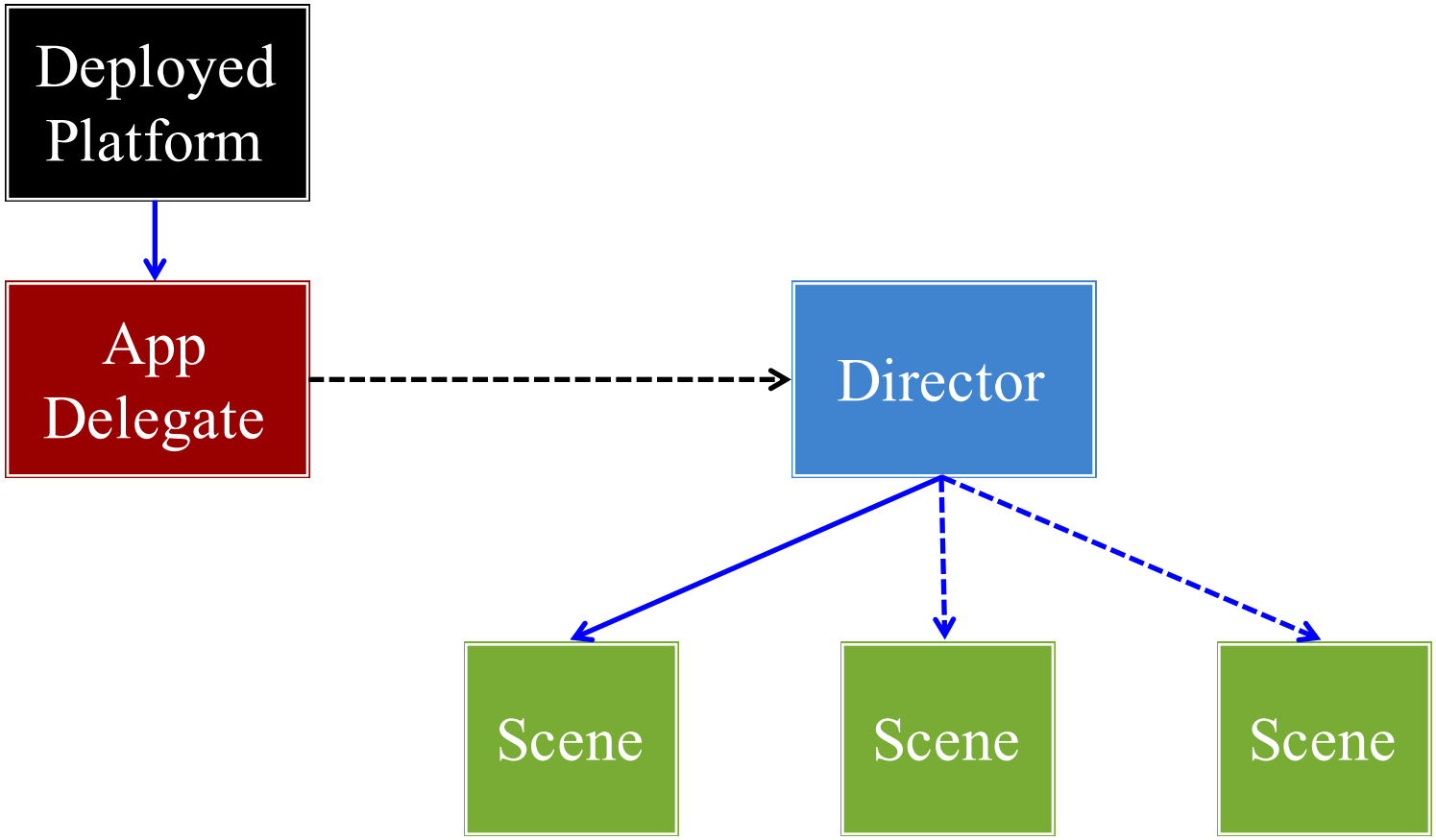
the gamedesigninitiative at cornell university

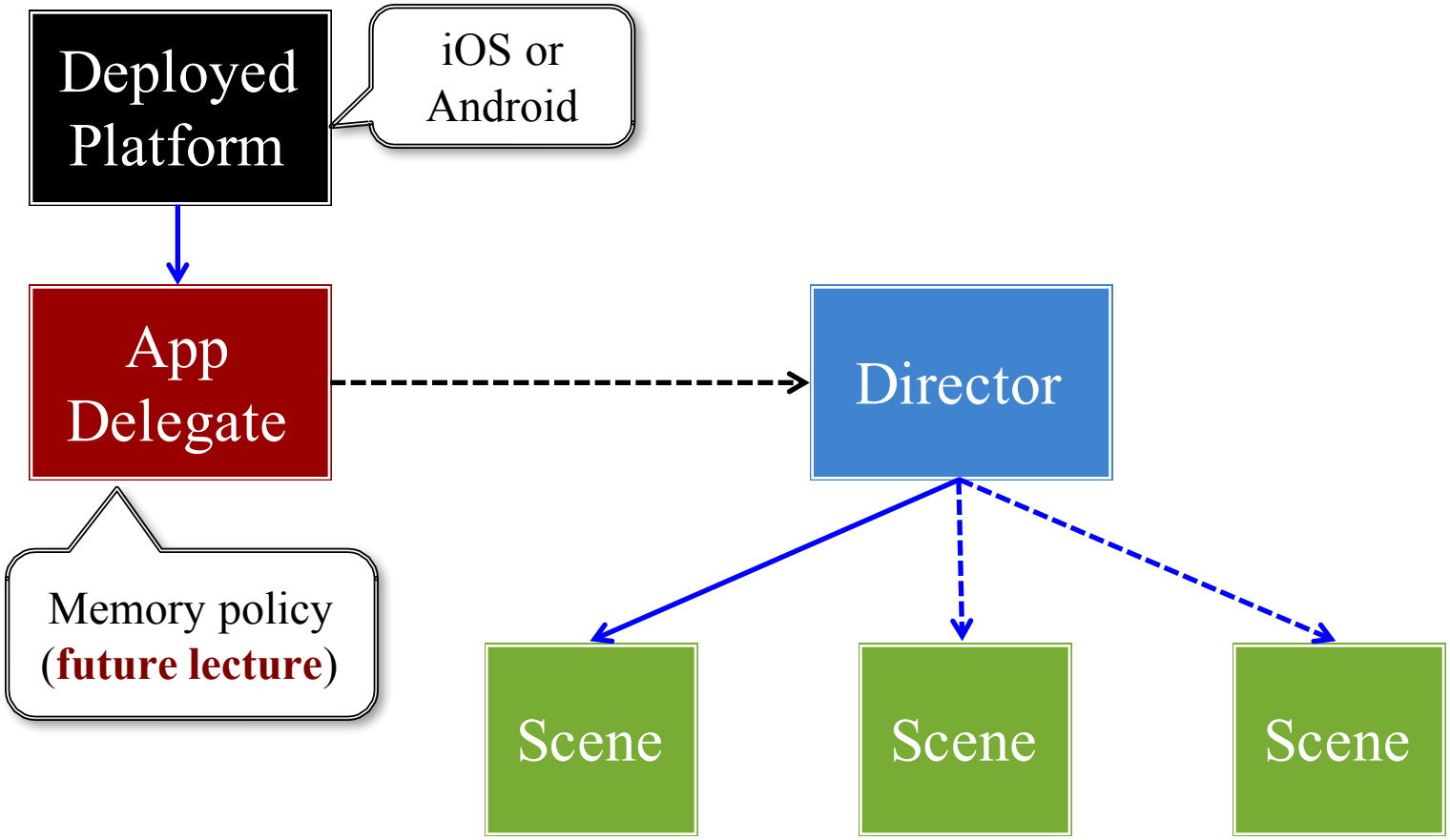
Lecture 9

Scene Graphs

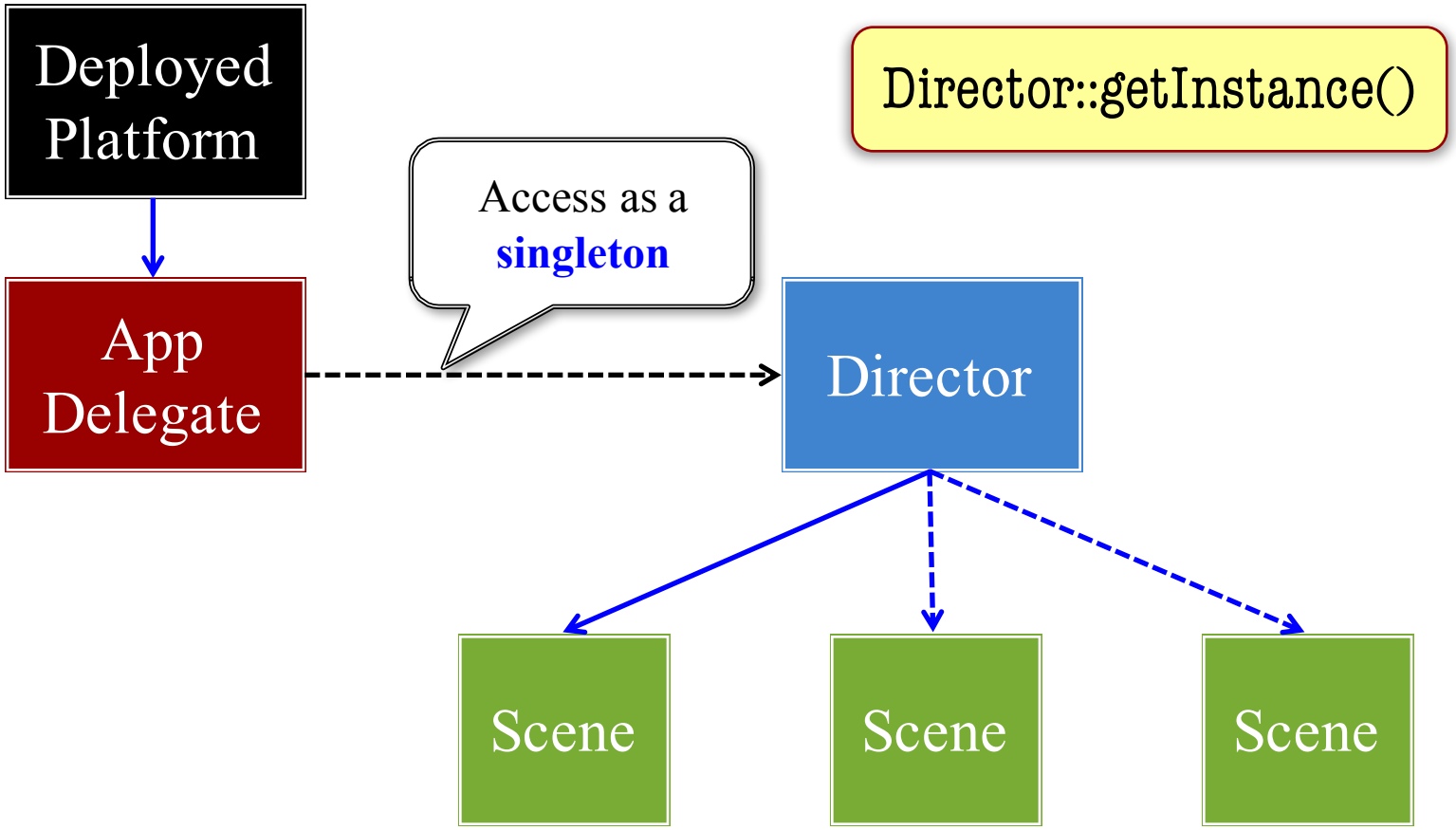
Structure of a Cocos2D-x Application



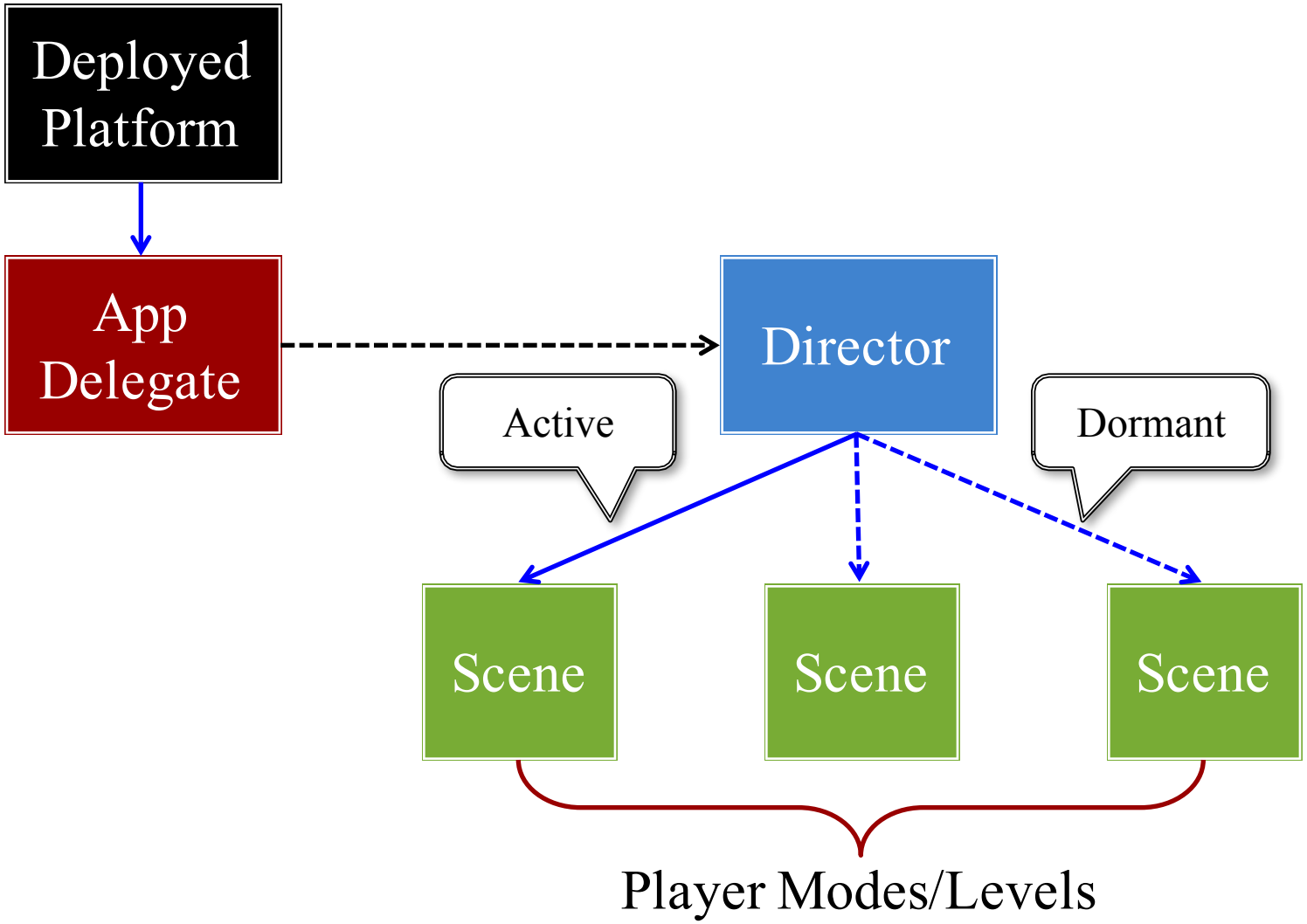
Structure of a Cocos2D-x Application



Structure of a Cocos2D-x Application

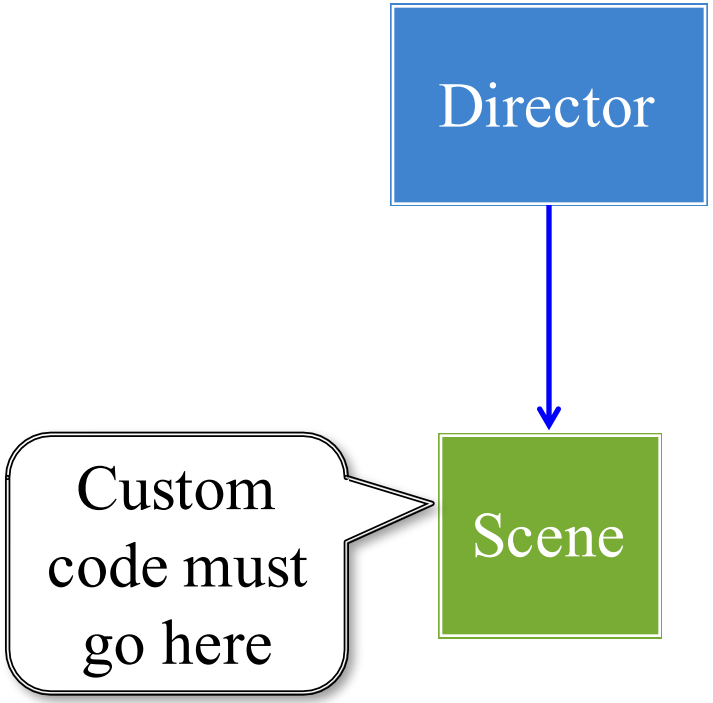


Structure of a Cocos2D-x Application

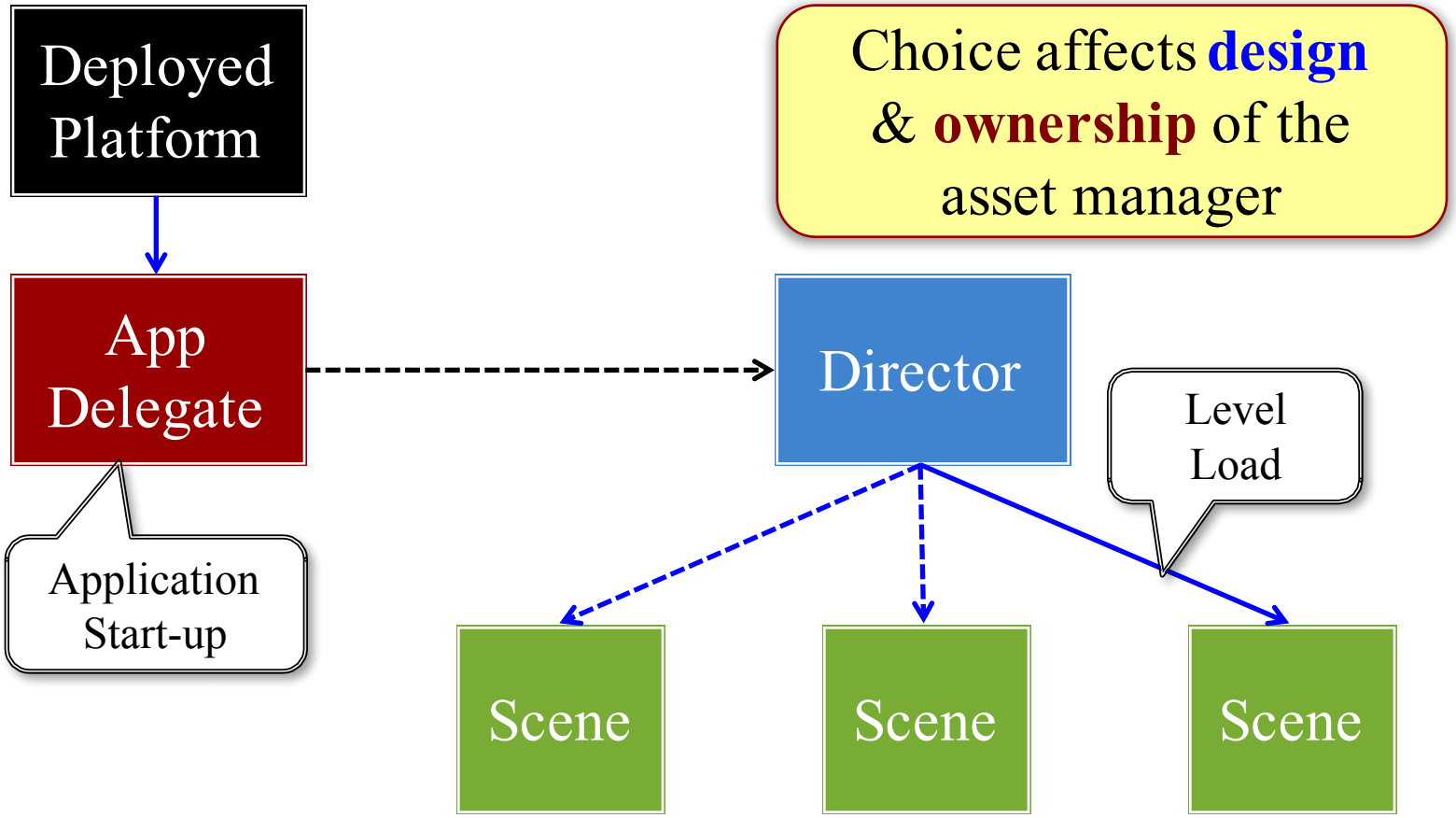


The Cocos2D-x Director

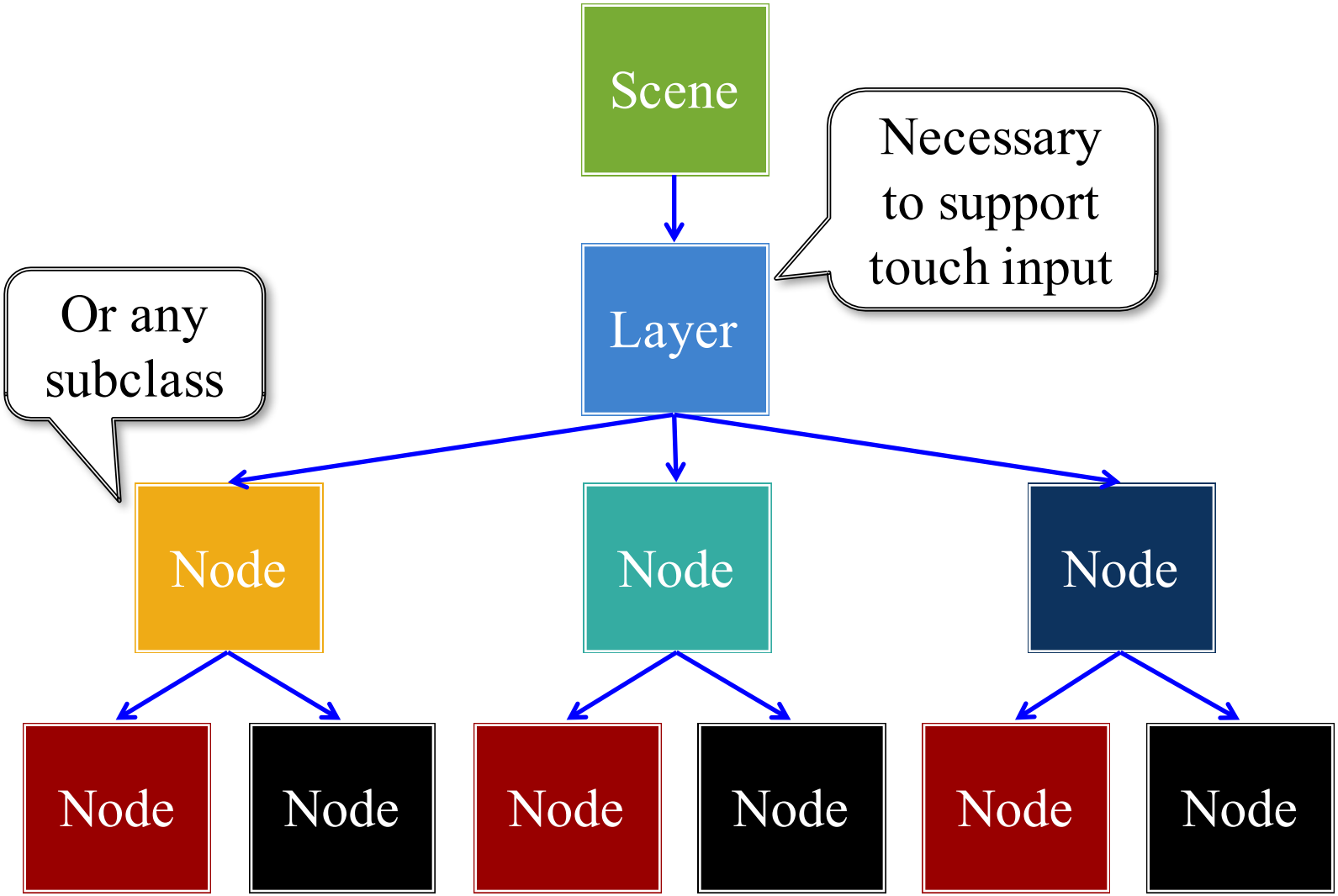
- Sort of like a controller
  - Manages main game loop
  - Effectivly owns everything
- It is a black-box **singleton**
  - You change settings…
  - but cannot add methods
- Adjustable settings
  - Current active scene
  - OpenGL drawing context
  - Cache of loaded textures
  - Input event listeners



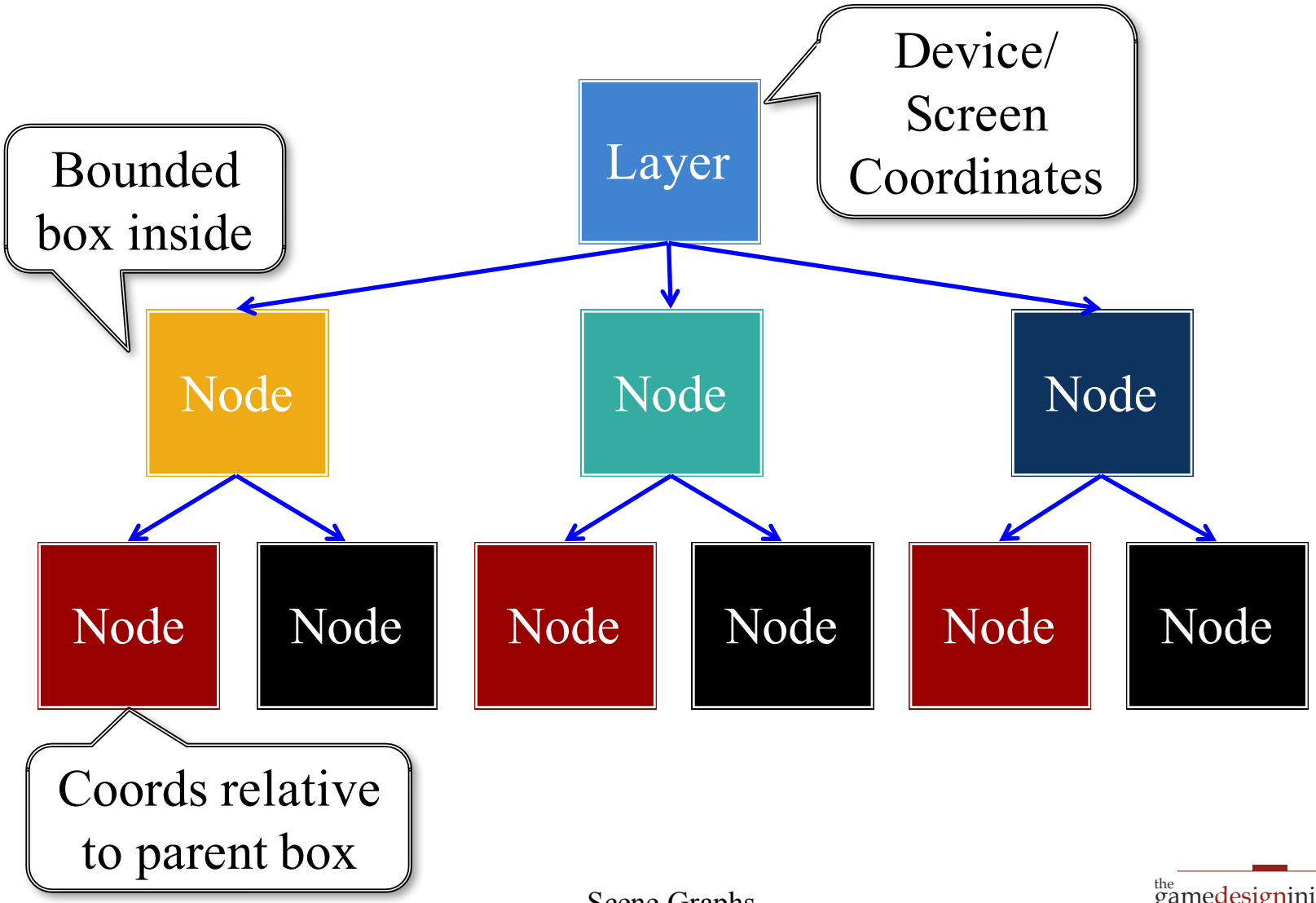
Aside: When Do We Load Assets?



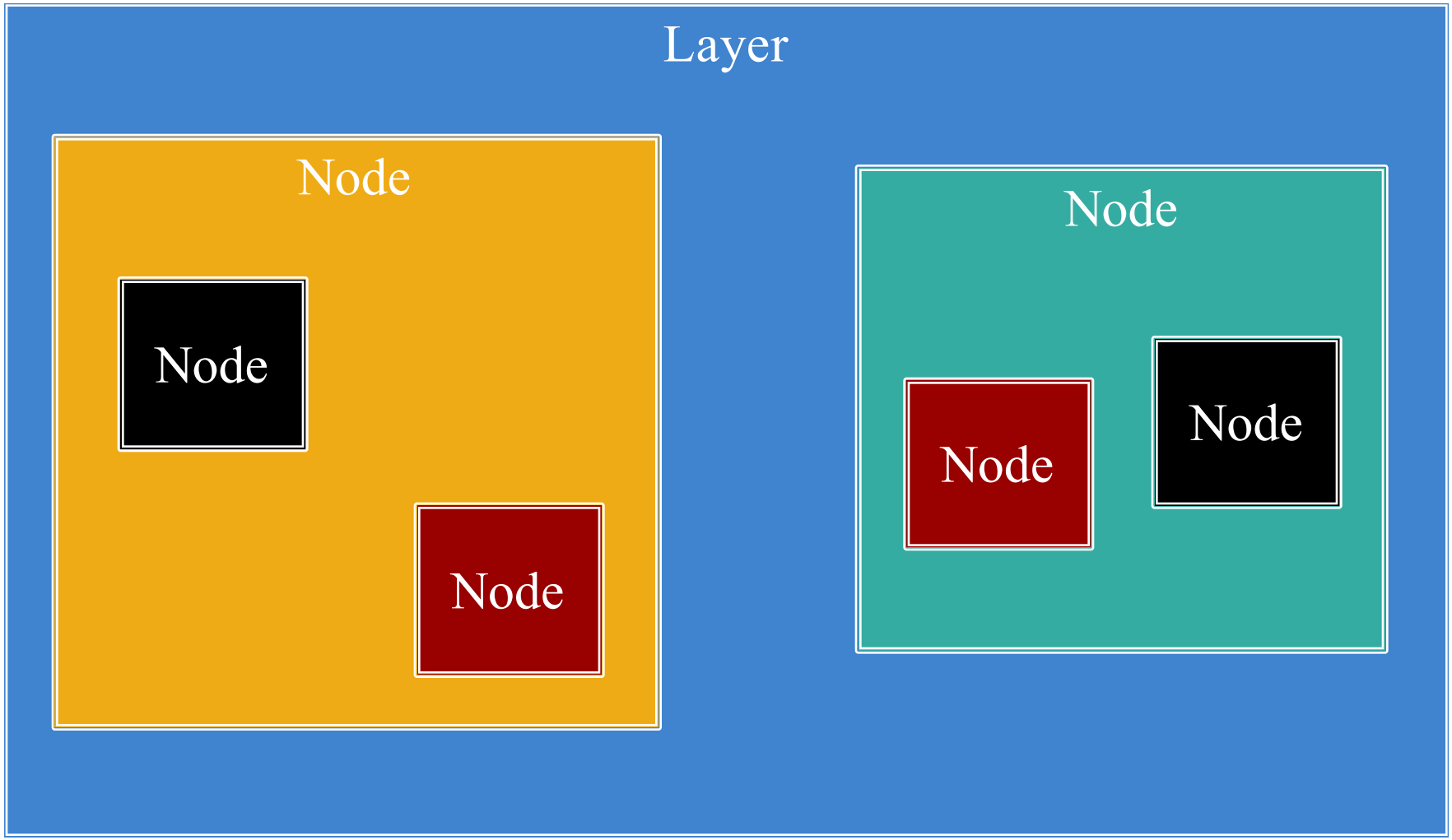
The Scene Graph



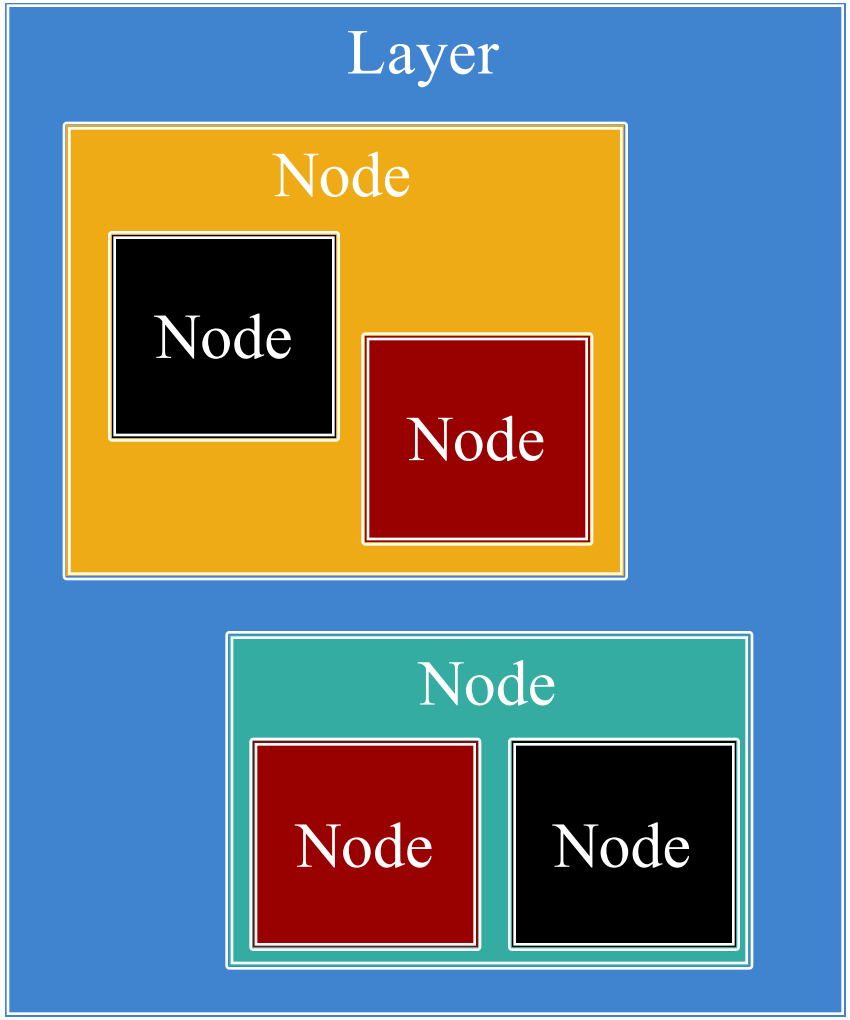
Each Node is a Coordinate System

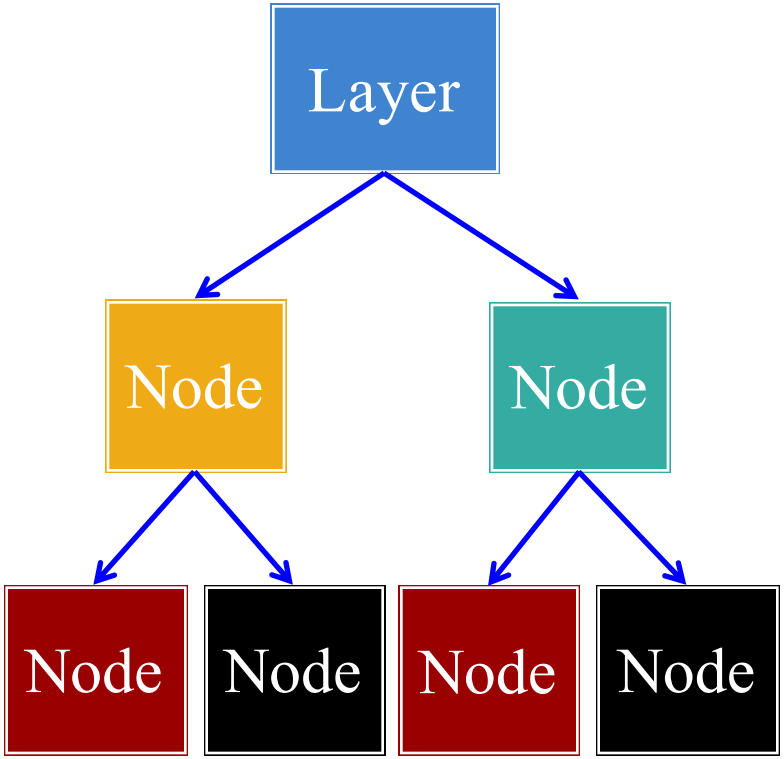


Each Node is a Coordinate System

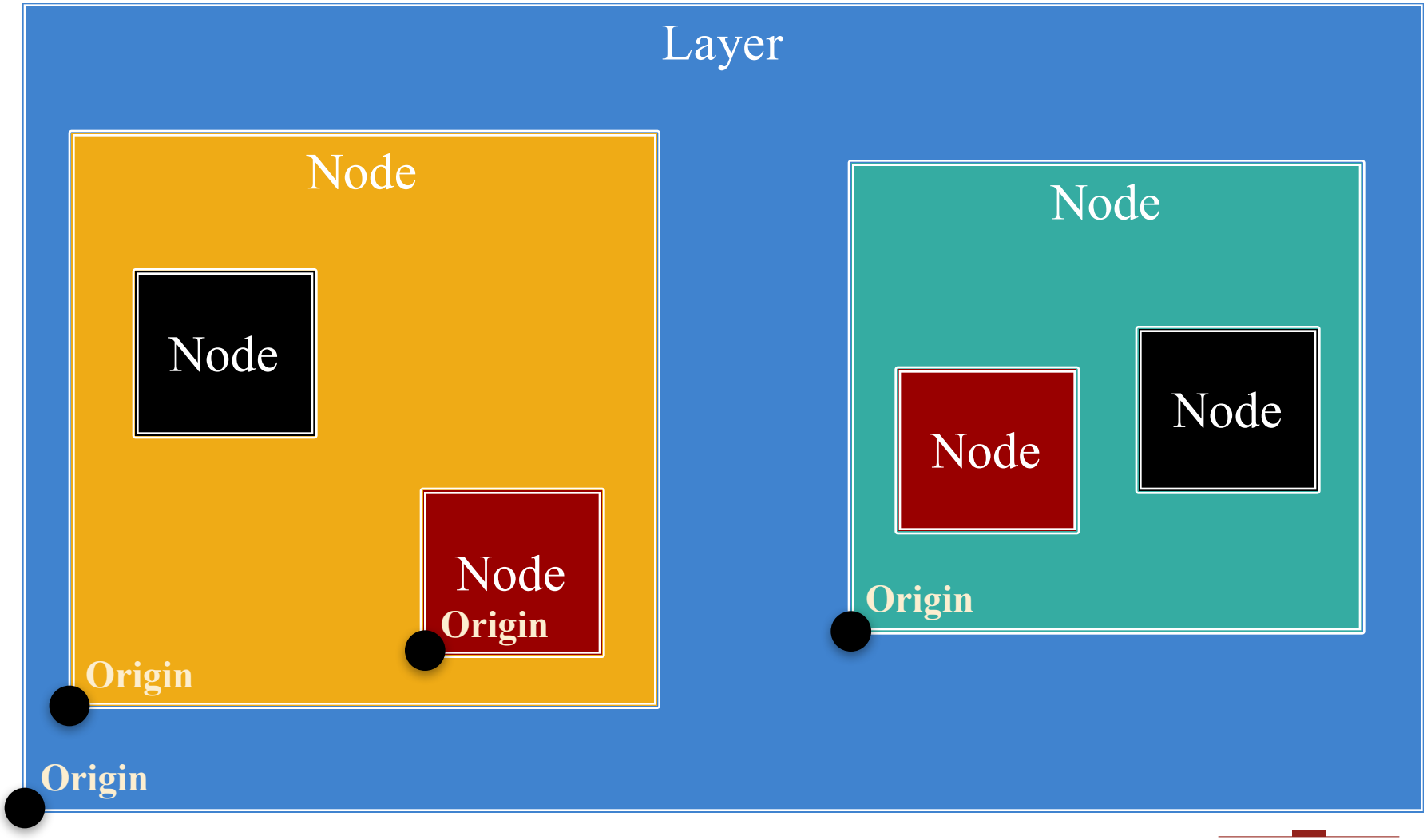


Each Node is a Coordinate System

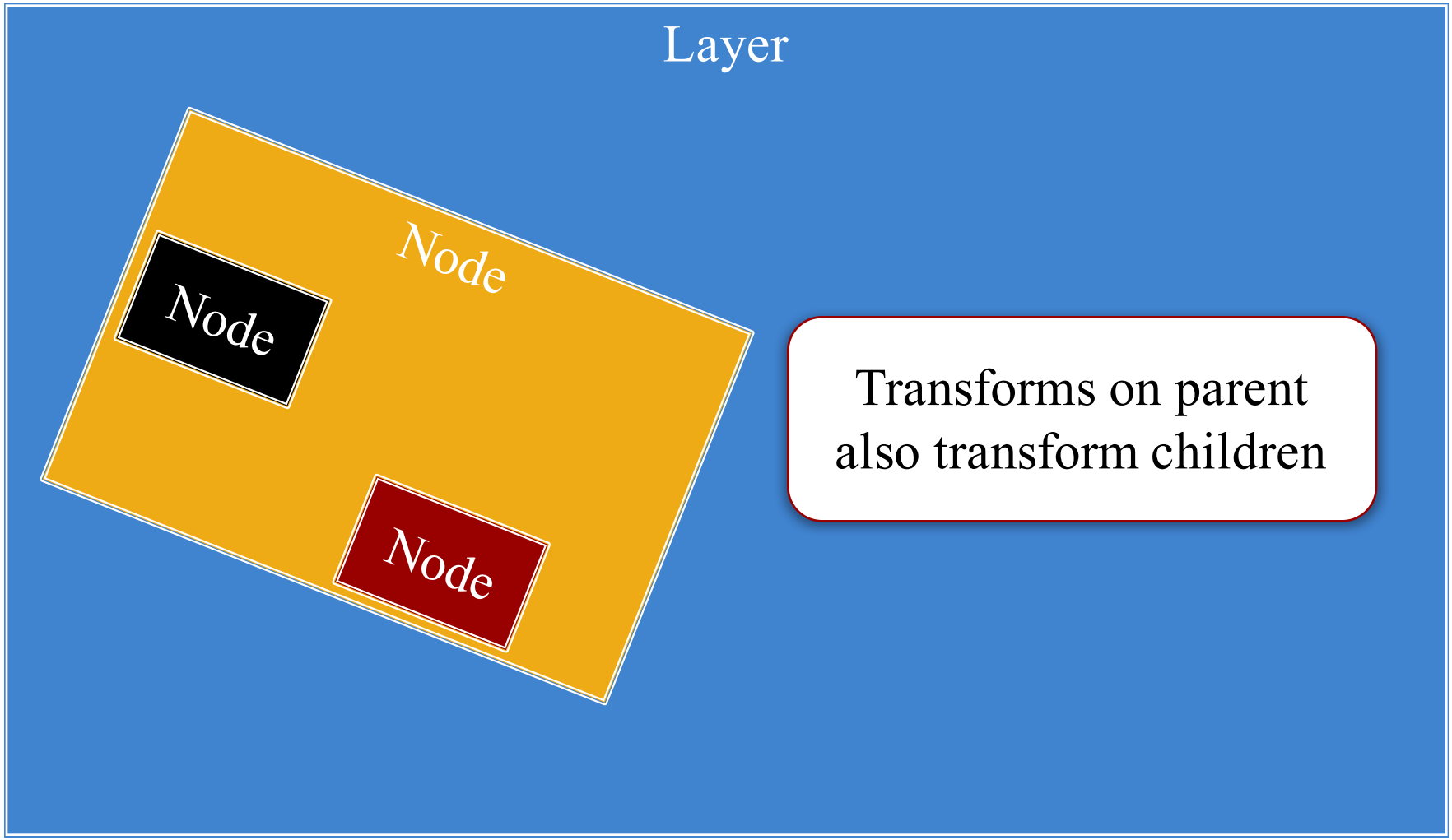




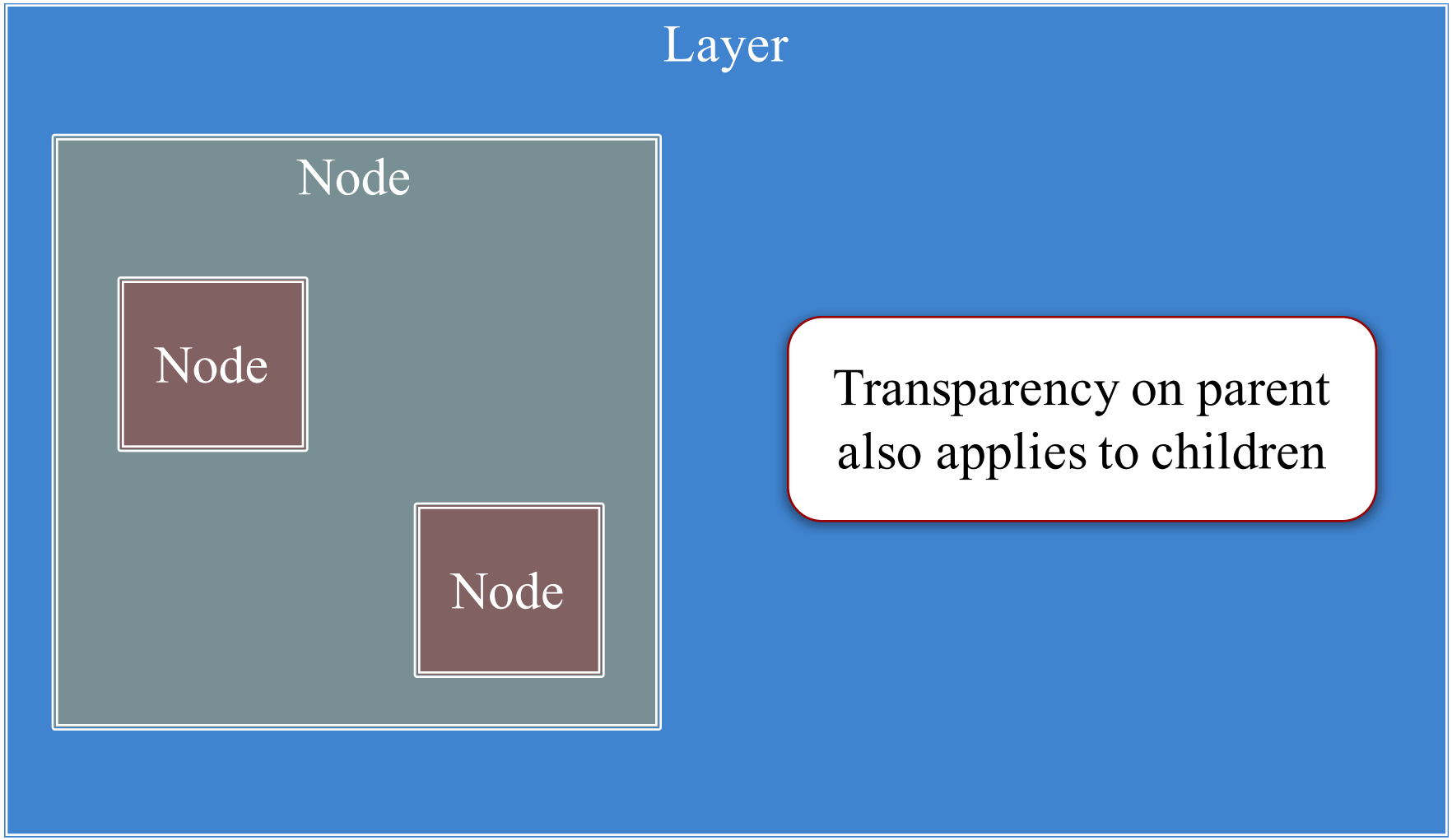
Each Node is a Coordinate System



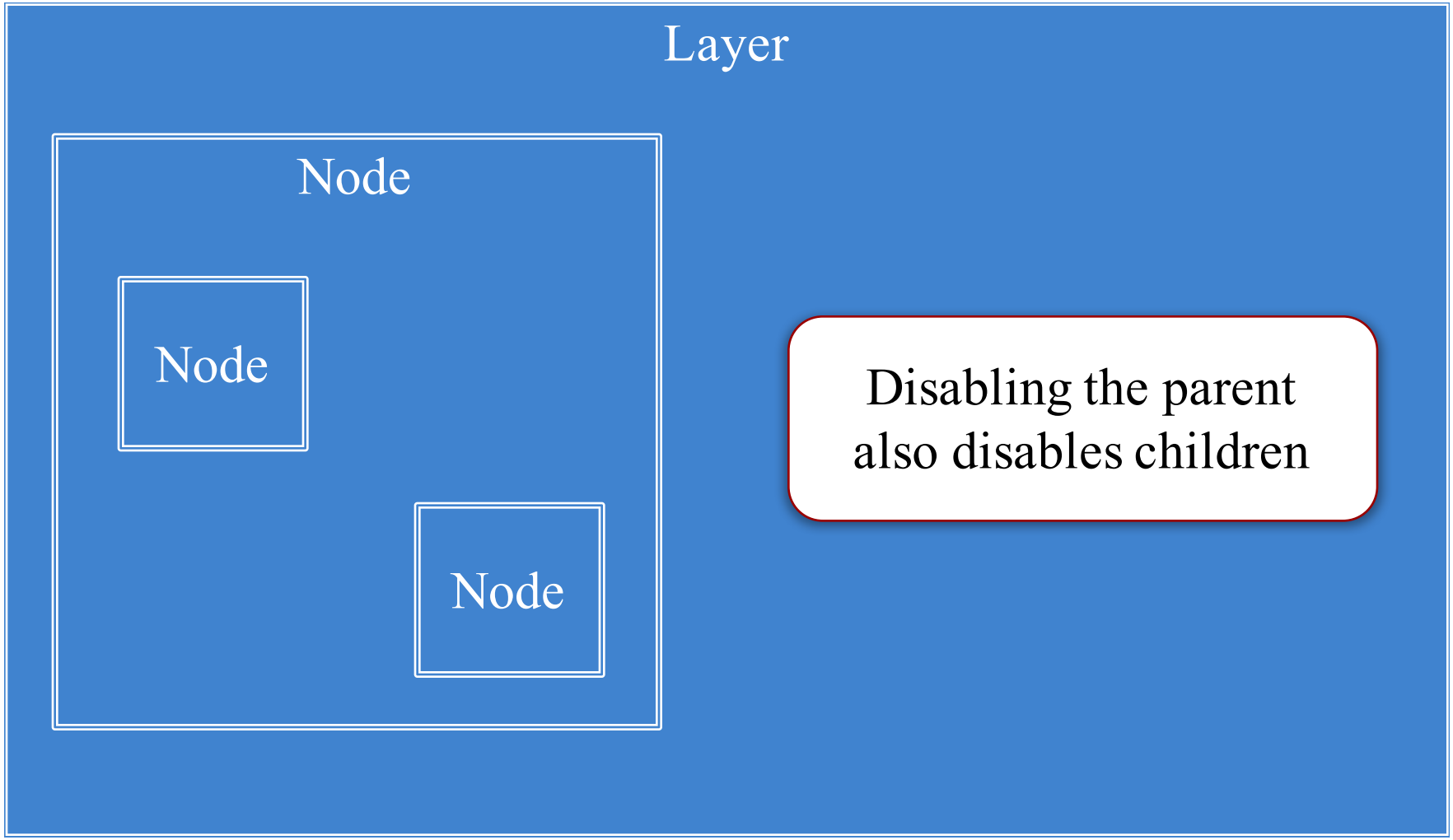
Settings Pass Down the Graph



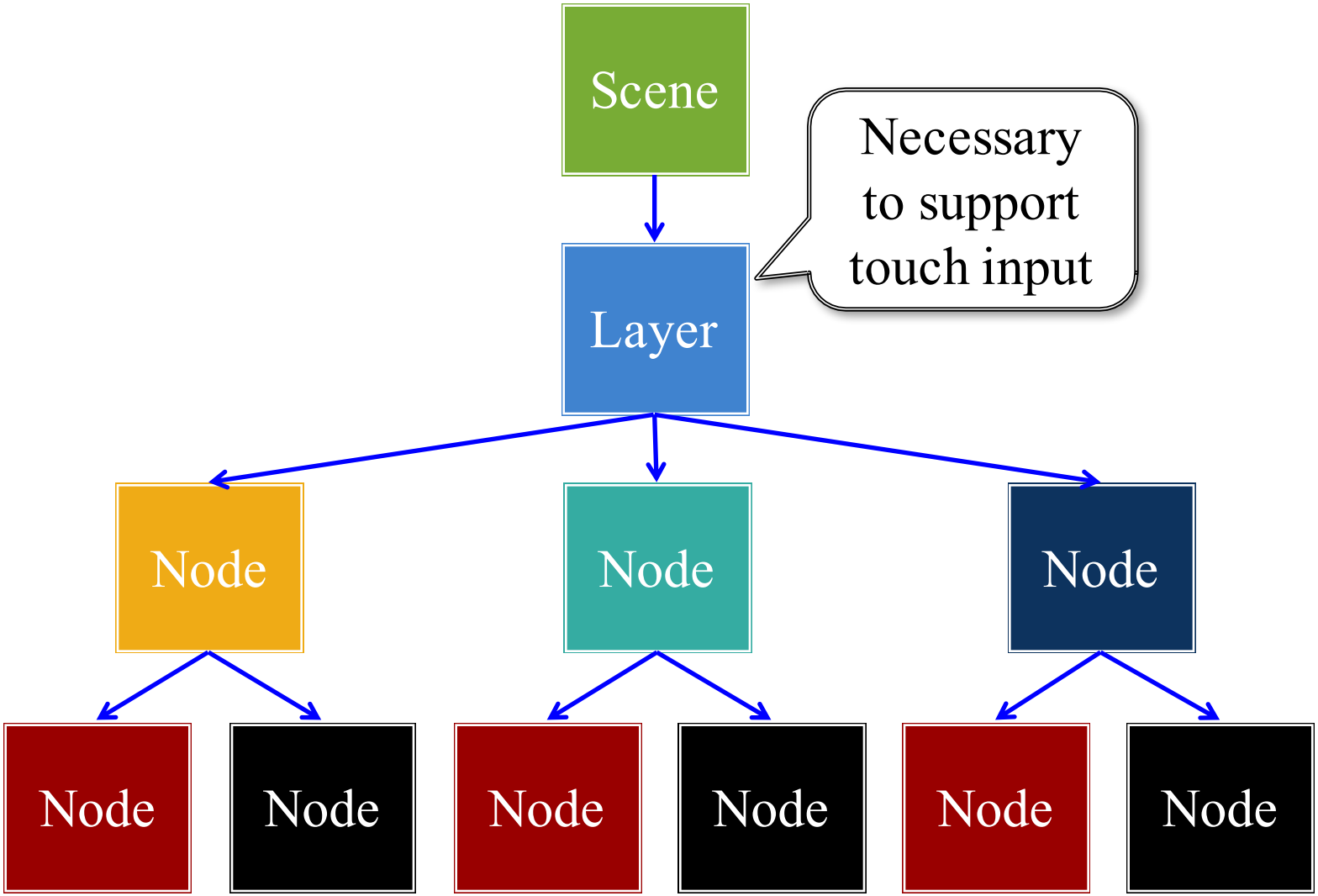
Settings Pass Down the Graph



Settings Pass Down the Graph



The Scene Graph



Creating a Custom Scene

- Engine has RootLayer class
  - Subclass of class Layer
  - Simplifies later subclassing
- Do not call its constructor!
  - Use function createScene
  - Call templated with class
  - **Example** (in AppDelegate): s = createScene<ShipRoot>()
  - In namespace GameRoot
- See the code samples
  - All follow this pattern

```
class GameMode: public RootLayer {
private:
    // Internal controllers/models
    ...

public:
    /** Starts the game */
    void start() override;

    /** Update one frame */
    void update(float dt) override;

    /** Stops the game */
    void stop() override;

};
```

The Two Main RootLayer Methods

`start()`

- Handles the game assets
  - Attaches the asset loaders
  - Loads immediate assets
- Starts any global singletons
  - **Example**: `SoundController`
- Creates any player modes
  - But does not launch ***yet***
  - Waits for assets to load
  - Like `GDXRoot` in 3152

`update()`

- Called each animation frame
- Manages gameplay
  - Converts input to actions
  - Processes NPC behavior
  - Resolves physics
  - Resolves other interactions
- Updates the scene graph
  - Transforms nodes
  - Enables/disables nodes

The Two Main RootLayer Methods

`start()`

- Handles the game assets
  - Attaches the asset loaders
  - Loads immediate assets
- S[illegible]tons
  - [illegible]ller
- C[illegible]ay player modes
  - But does not launch ***yet***
  - Waits for assets to load
  - Like `GDXRoot` in 3152

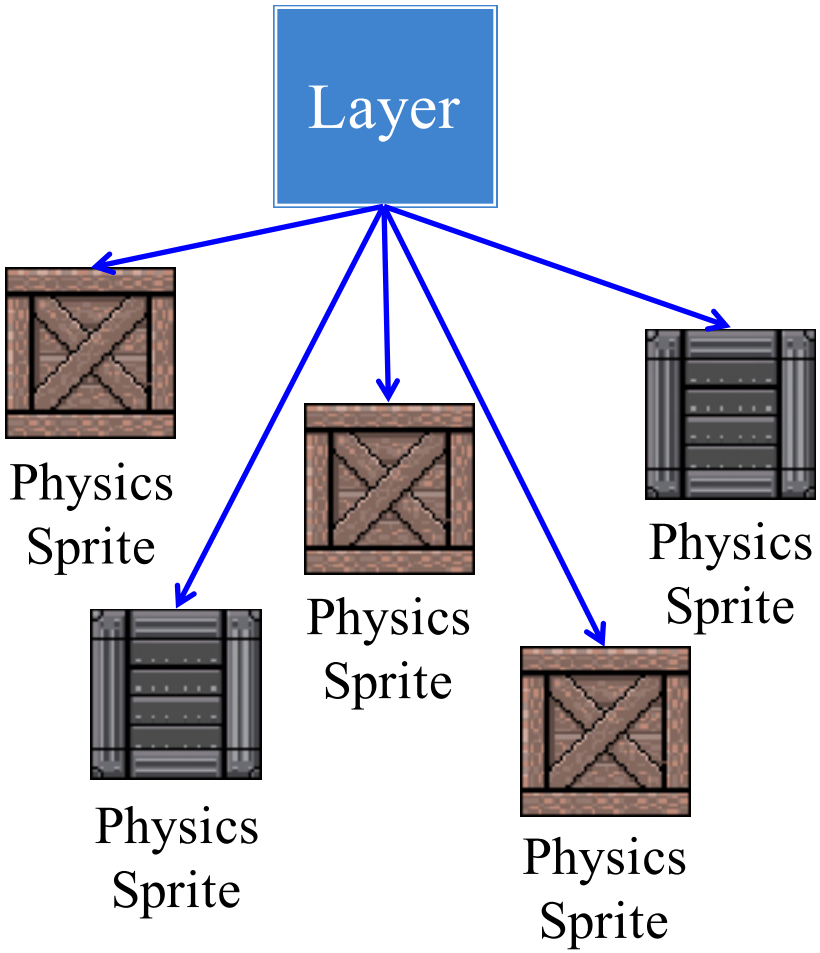
stop() just cleans up after **start()**

`update()`

- Called each animation frame
- Manages gameplay
  - Converts in[illegible]ons
  - [illegible]or
  - [illegible]
  - [illegible]solves other interactions
- Updates the scene graph
  - Transforms nodes
  - Enables/disables nodes

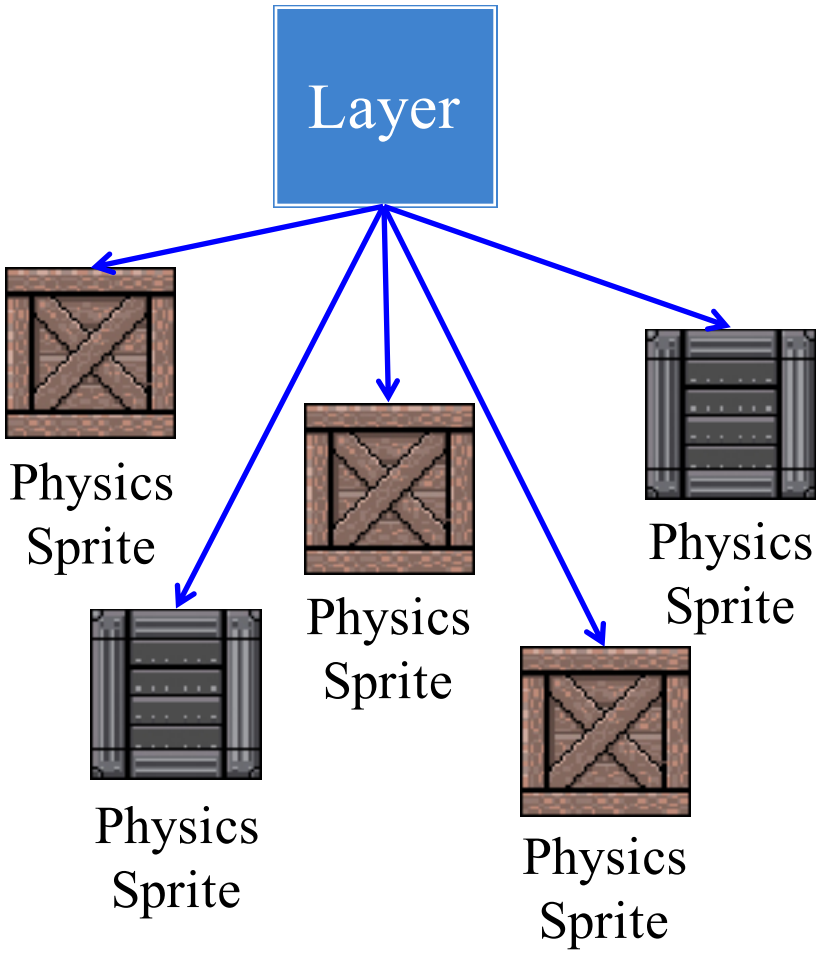
Does not draw! Handled separately

The Cocos2D-x Philosophy



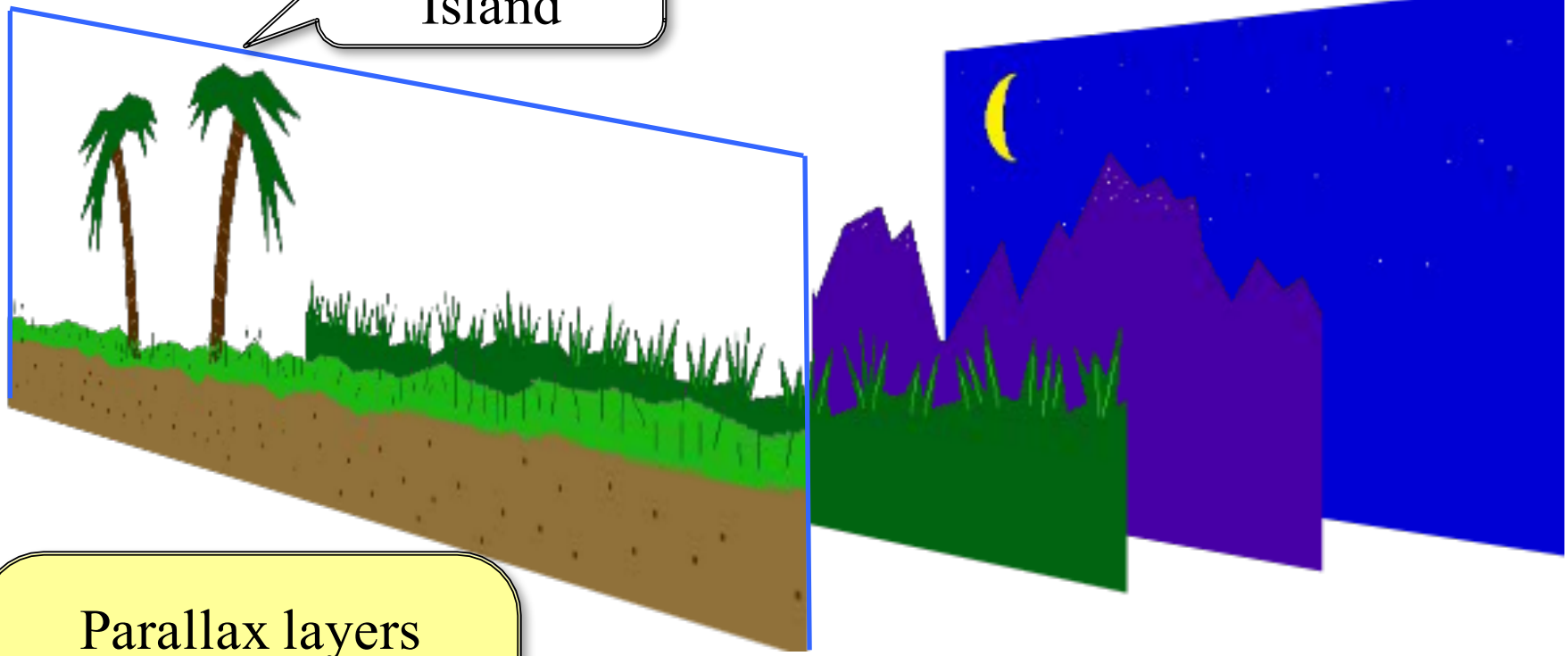
- Each model is its own node
  - Node coords = texture coords
  - Model pos = node pos
- Each node is a Box2D body
  - Scene is a Box2D world
  - Nodes moved automatically
- Each node processes input
  - Node has custom listener
  - Gets touches on that node
- **Massive violation of MVC**

The Cocos2D Unity3D Philosophy



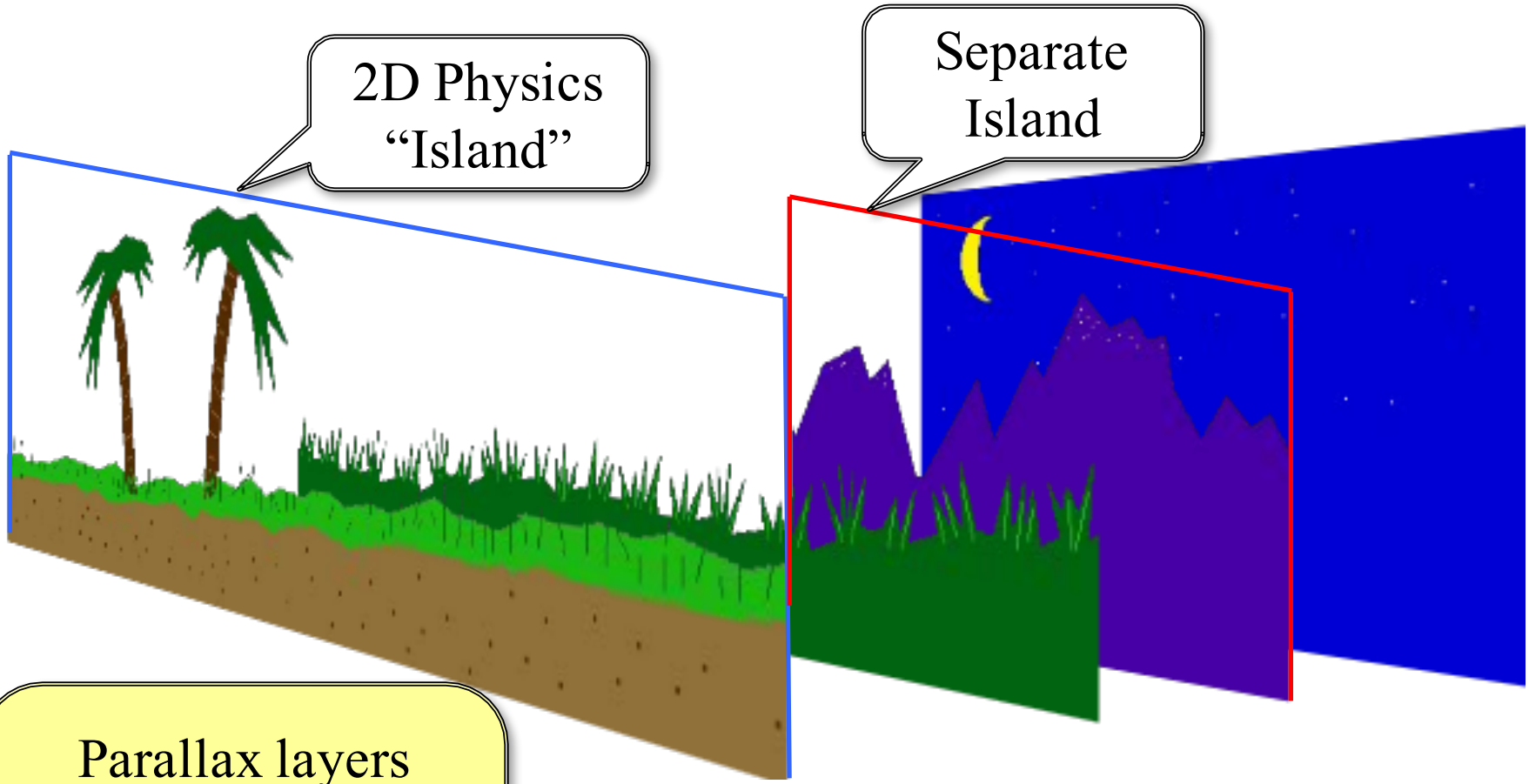
- Each model is its own node
  - Node coords = texture coords
  - Model pos = node pos
- Each node is a Box2D body
  - Scene is a Box2D world
  - Nodes moved automatically
- Each node processes input
  - Node has custom listener
  - Gets touches on that node
- **Massive violation of MVC**

The Problem: **Physics**



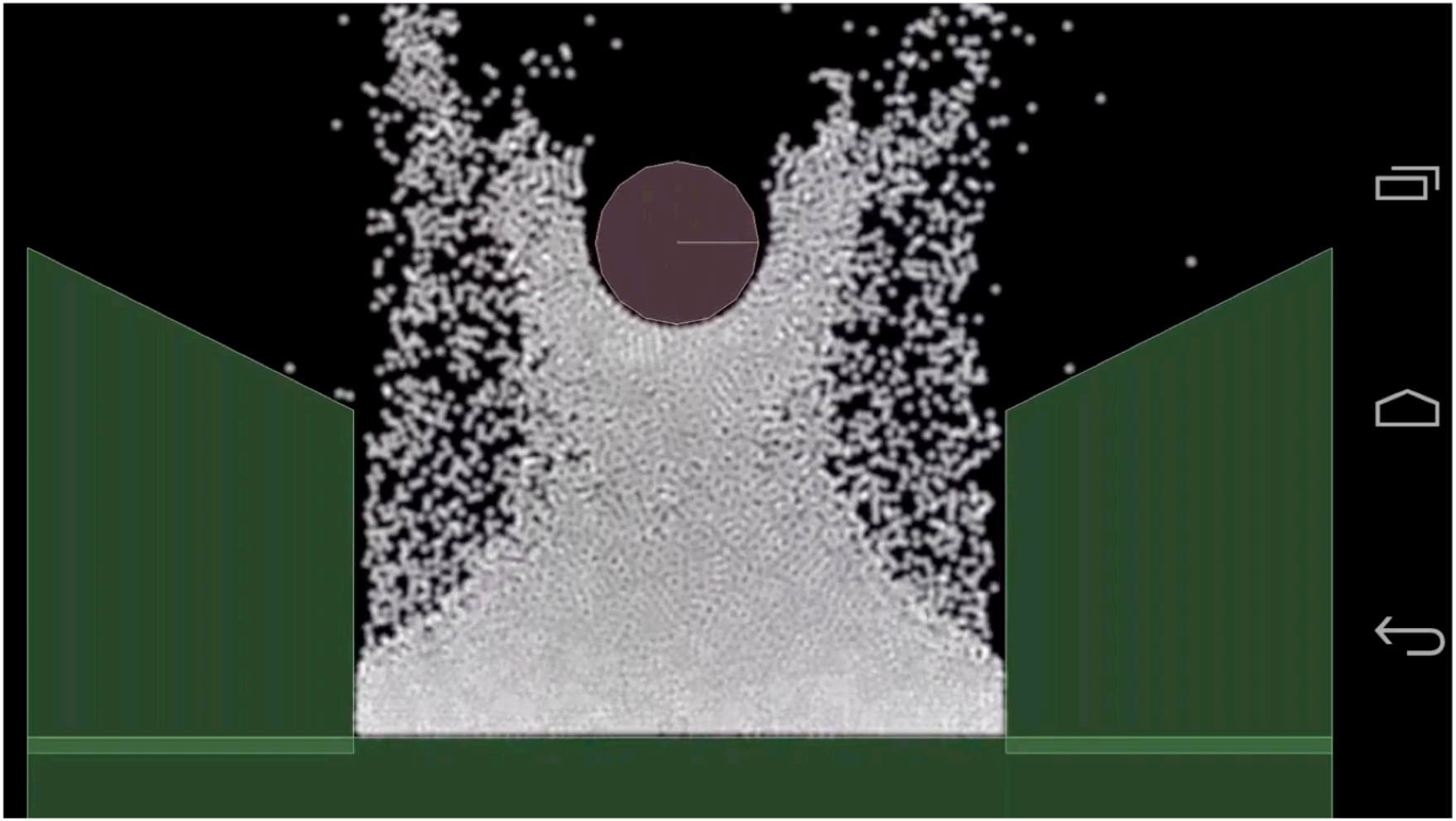
Parallax layers
(e.g. beat-em up)

The Problem: **Physics**

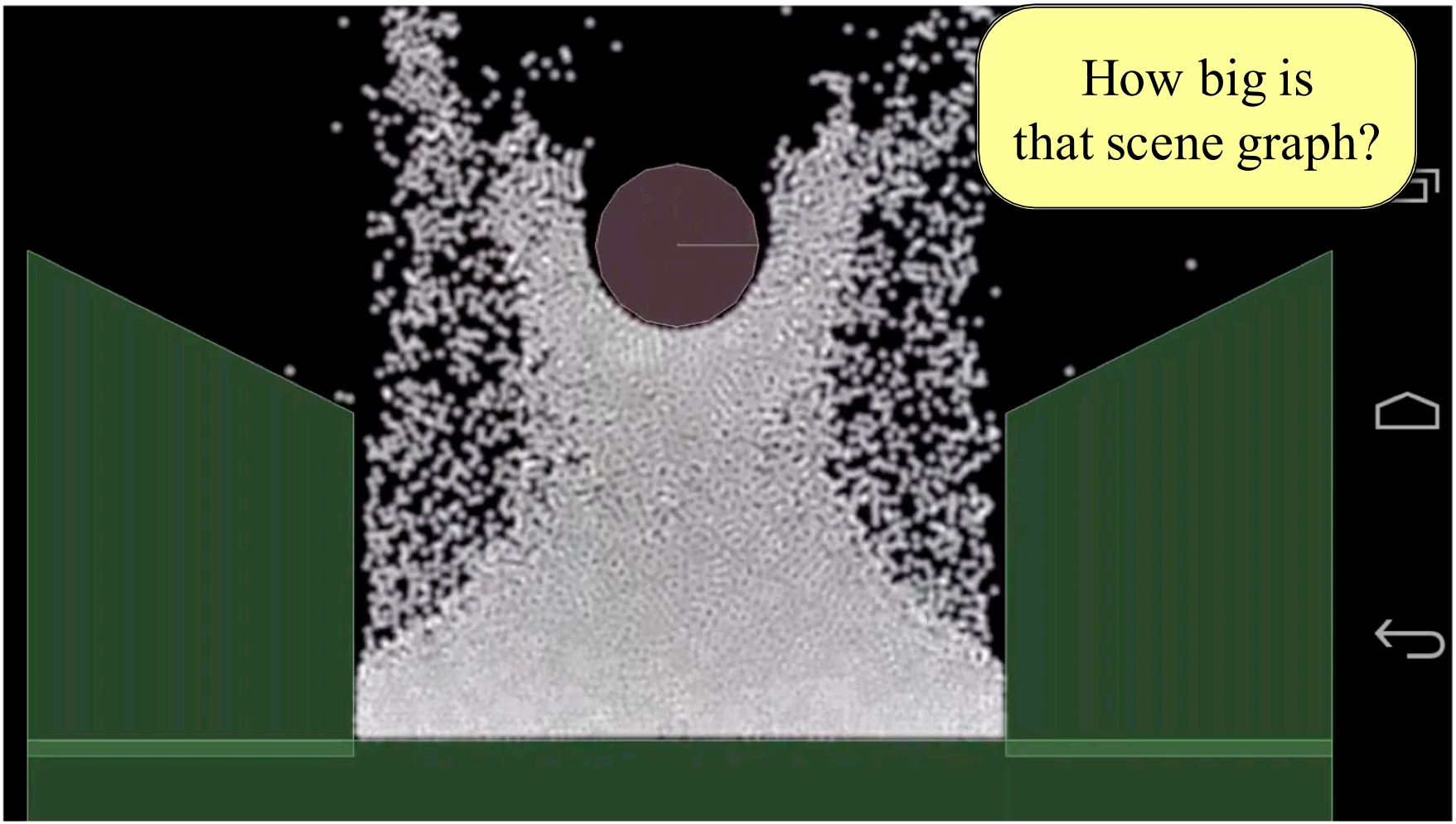


Parallax layers
(e.g. beat-em up)

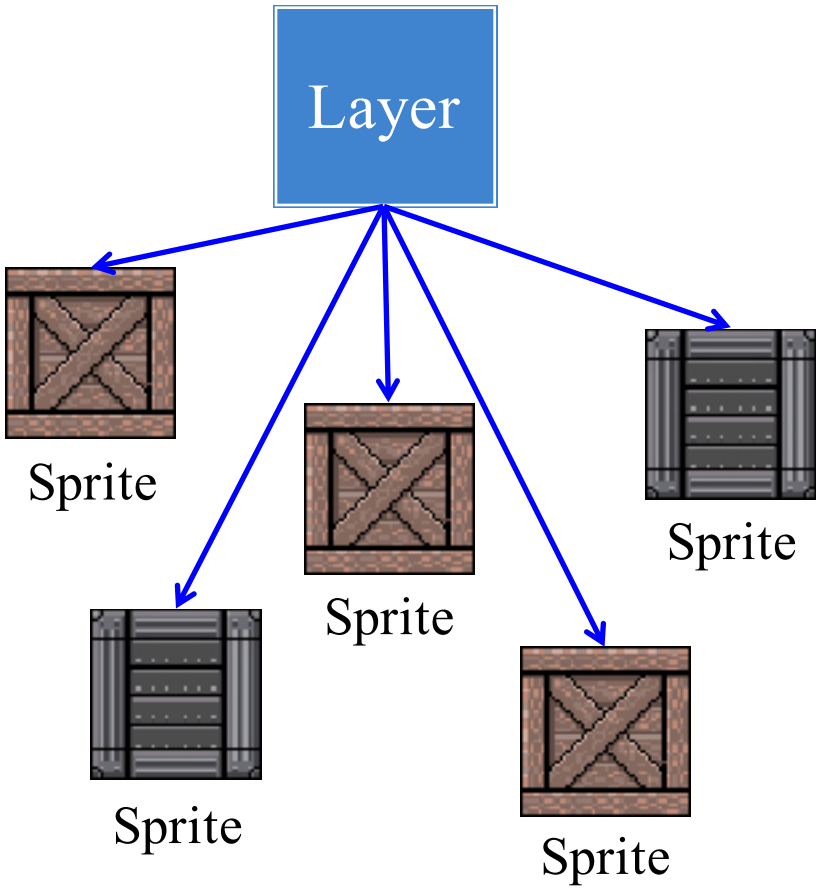
The Problem: **Physics**



The Problem: **Physics**

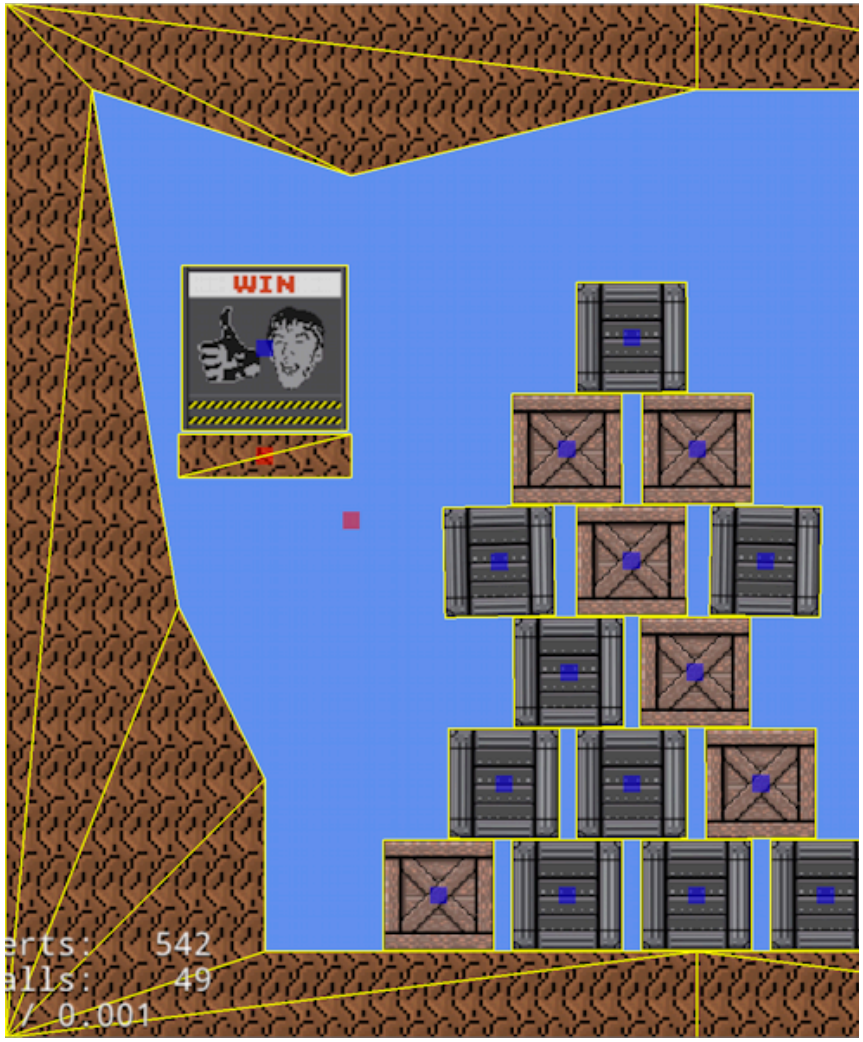


Bigger Problem: **Rendering**



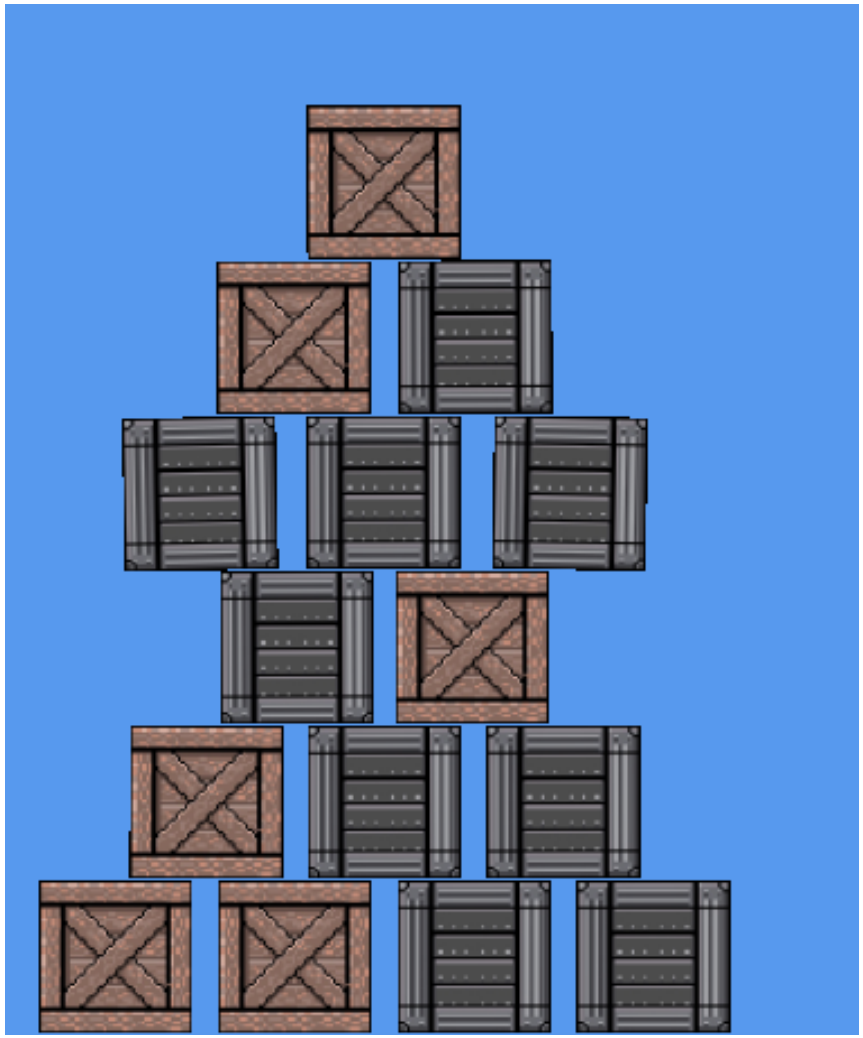
- Cocos2D 2.x Rendering
  - Node = new graphics call
  - Even if the same texture
  - Many calls on small data
- GPU cards do not like this
  - Each call requires card I/O
  - Want few calls on large data
  - Else performance stalls
- **Result**: Horrible framerate
  - Cocos2D 3.x changed this

What Do We Really Want?



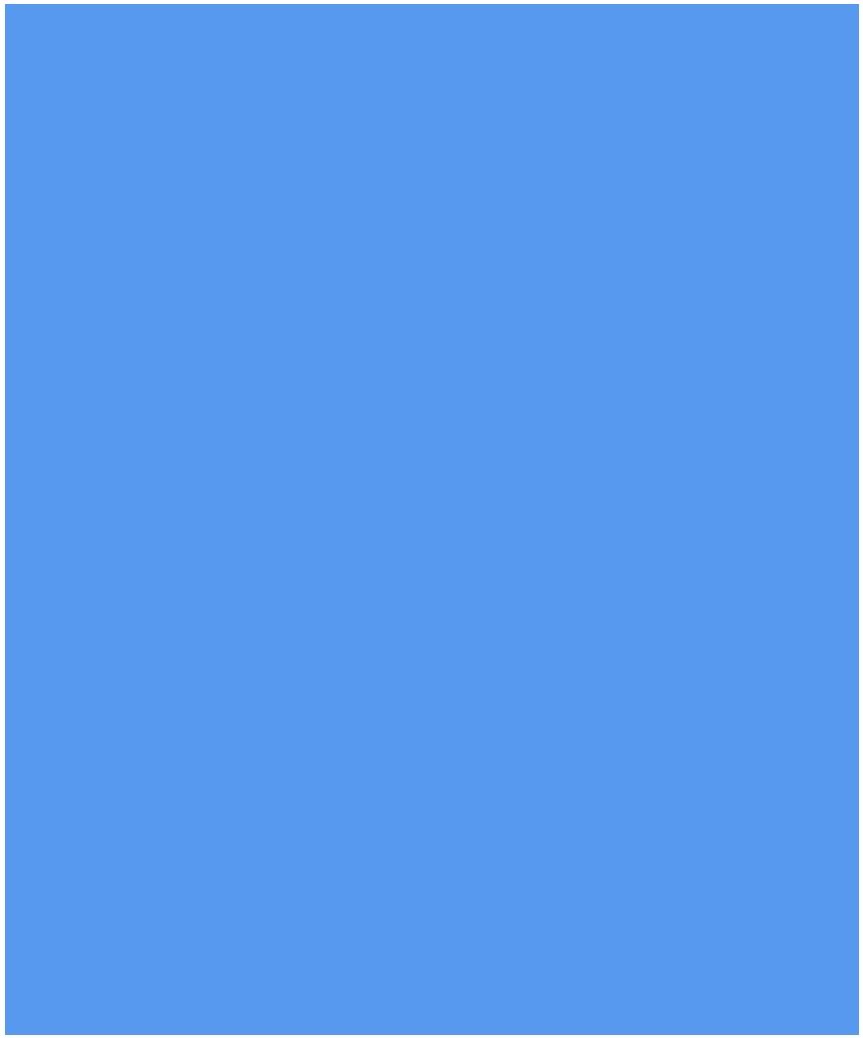
- Sprites = textured **triangles**
  - Gather all sprite vertices
  - Make one list of triangles
  - Send them to GPU at once
- But stall on texture change
  - Reorder data on texture
  - Limits texture switches
  - Safe if there is **no overlap**
- Is there a name for this?

What Do We Really Want?



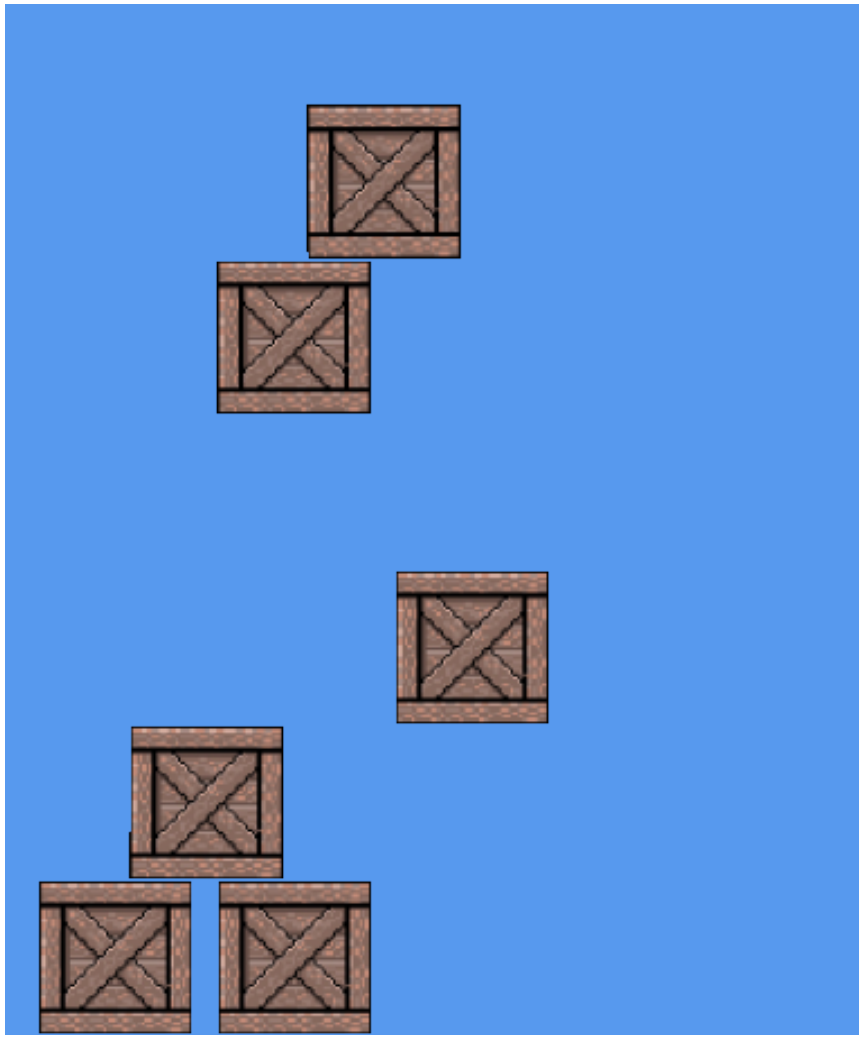
- Sprites = textured **triangles**
  - Gather all sprite vertices
  - Make one list of triangles
  - Send them to GPU at once
- But stall on texture change
  - Reorder data on texture
  - Limits texture switches
  - Safe if there is **no overlap**
- Is there a name for this?

What Do We Really Want?



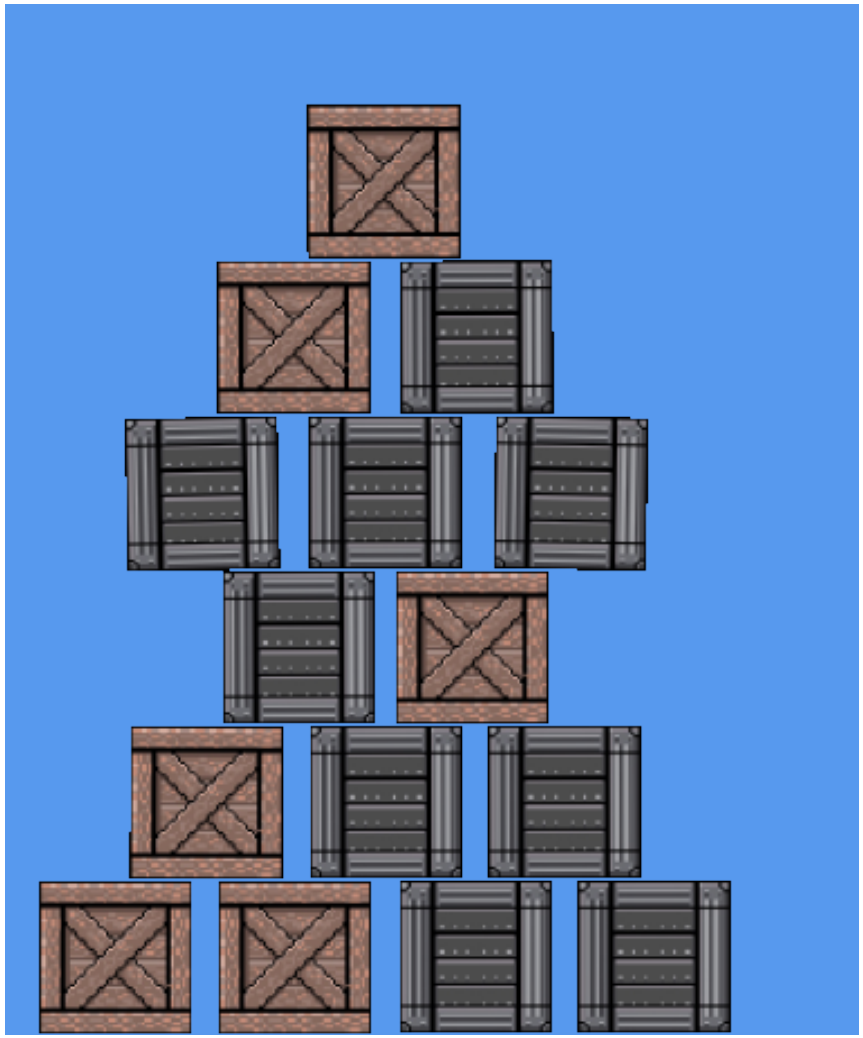
- Sprites = textured **triangles**
  - Gather all sprite vertices
  - Make one list of triangles
  - Send them to GPU at once
- But stall on texture change
  - Reorder data on texture
  - Limits texture switches
  - Safe if there is **no overlap**
- Is there a name for this?

What Do We Really Want?



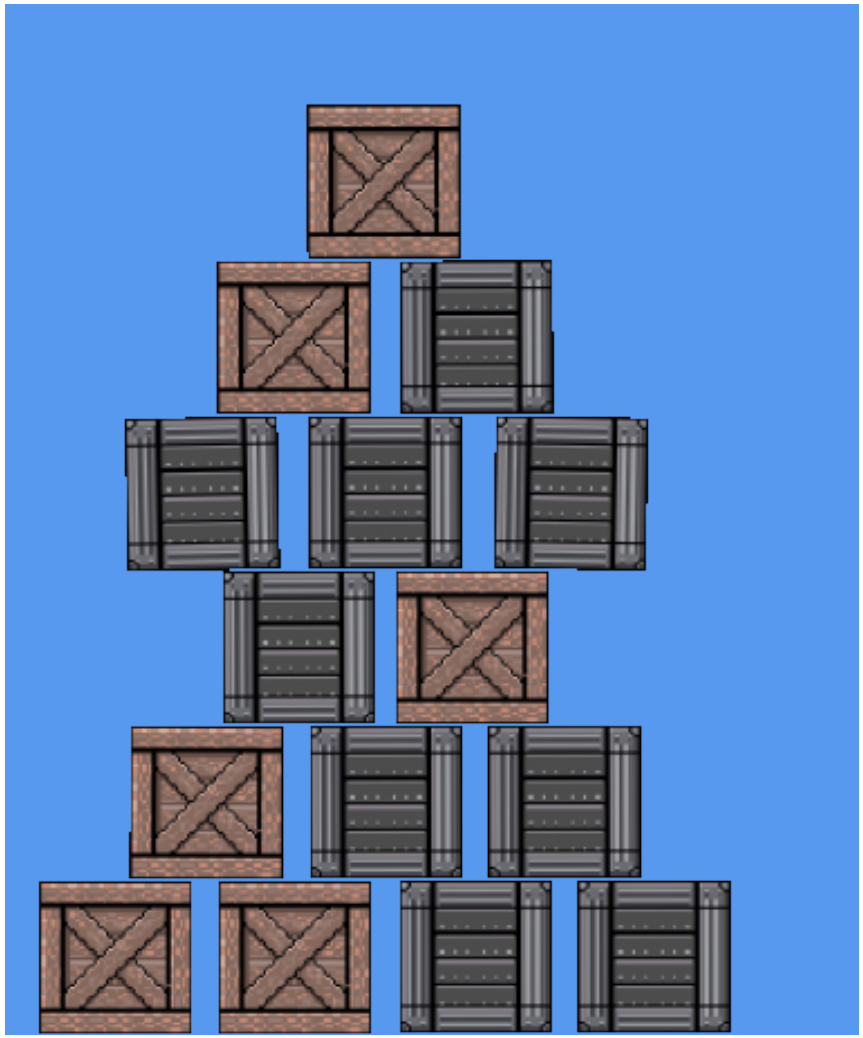
- Sprites = textured **triangles**
  - Gather all sprite vertices
  - Make one list of triangles
  - Send them to GPU at once
- But stall on texture change
  - Reorder data on texture
  - Limits texture switches
  - Safe if there is **no overlap**
- Is there a name for this?

What Do We Really Want?



- Sprites = textured **triangles**
  - Gather all sprite vertices
  - Make one list of triangles
  - Send them to GPU at once
- But stall on texture change
  - Reorder data on texture
  - Limits texture switches
  - Safe if there is **no overlap**
- Is there a name for this?

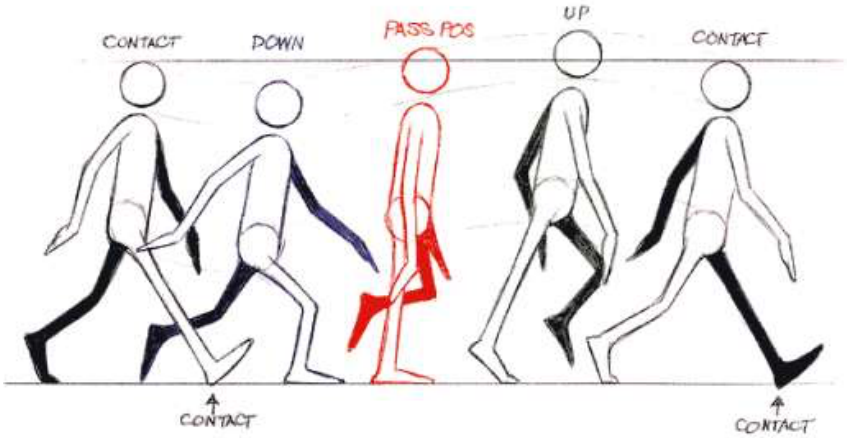
What Do We Really Want?



- Sprites = textured **triangles**
  - Gather all sprite vertices
  - Make one list of triangles
  - Send them to GPU at once
- But stall on texture change
  - Reorder data on texture
  - Limits texture switches
  - Safe if there is **no overlap**
- Is there a name for this?
  - **SpriteBatch**!

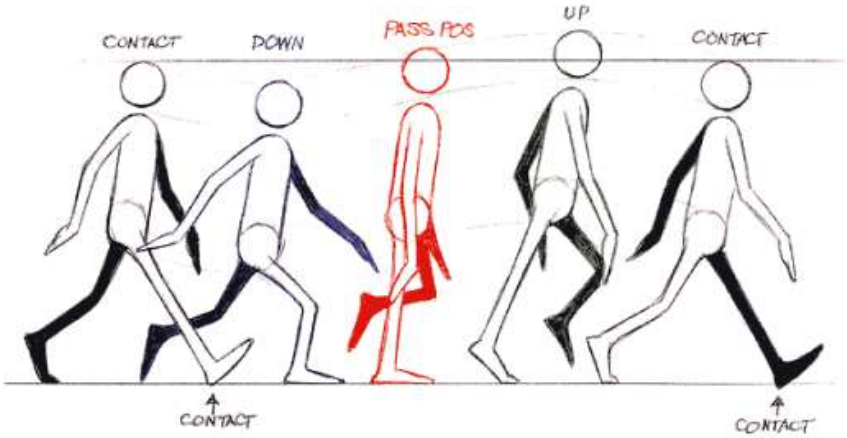
So Why Use a Scene Graph?

- **Animation** is much easier!
- Can reduce filmstrips
  - Break asset into parts
  - Each has a coord system
  - Transform each separately
- Decouple animation loop
  - Update does not set frame
  - Node advances frame
  - Update switches animations
- **Exception**: ShipDemo



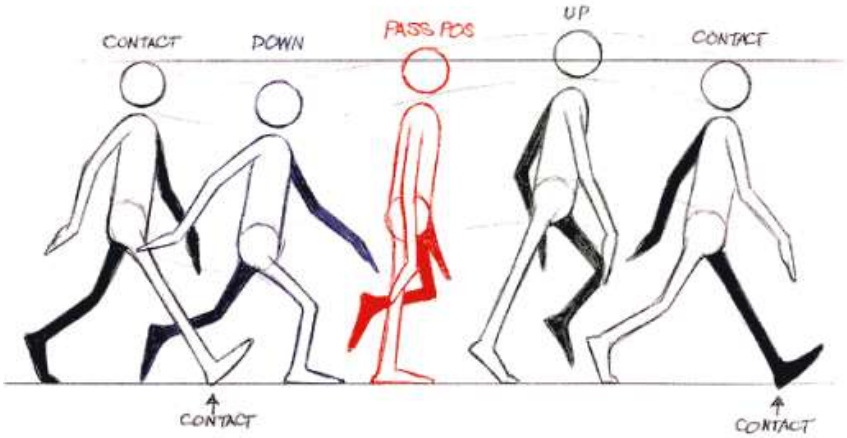
So Why Use a Scene Graph?

- **Animation** is much easier!
- Can reduce filmstrips
  - Break asset into parts
  - Each has a coord system
  - Transform each separately
- Decouple animation loop
  - Update does not set frame
  - Node advances frame
  - Update switches animations
- **Exception**: ShipDemo



So Why Use a Scene Graph?

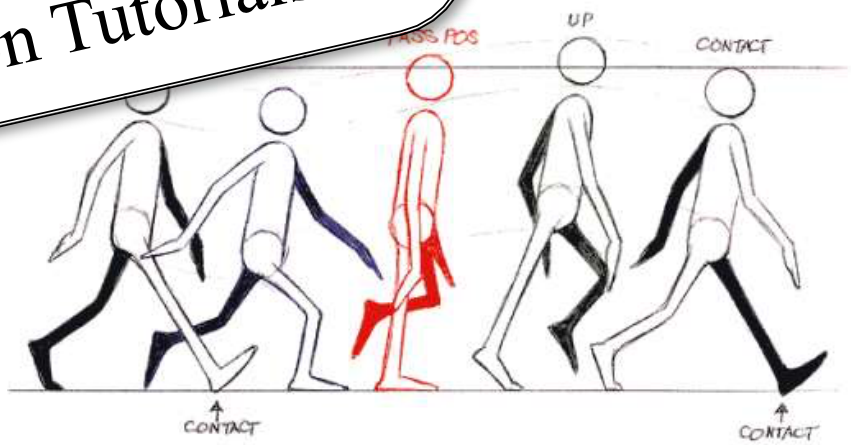
- **Animation** is much easier!
- Can reduce filmstrips
  - Break asset into parts
  - Each has a coord system
  - Transform each separately
- Decouple animation loop
  - Update does not set frame
  - Node advances frame
  - Update switches animations
- **Exception**: ShipDemo



So Why Use a Scene Graph?

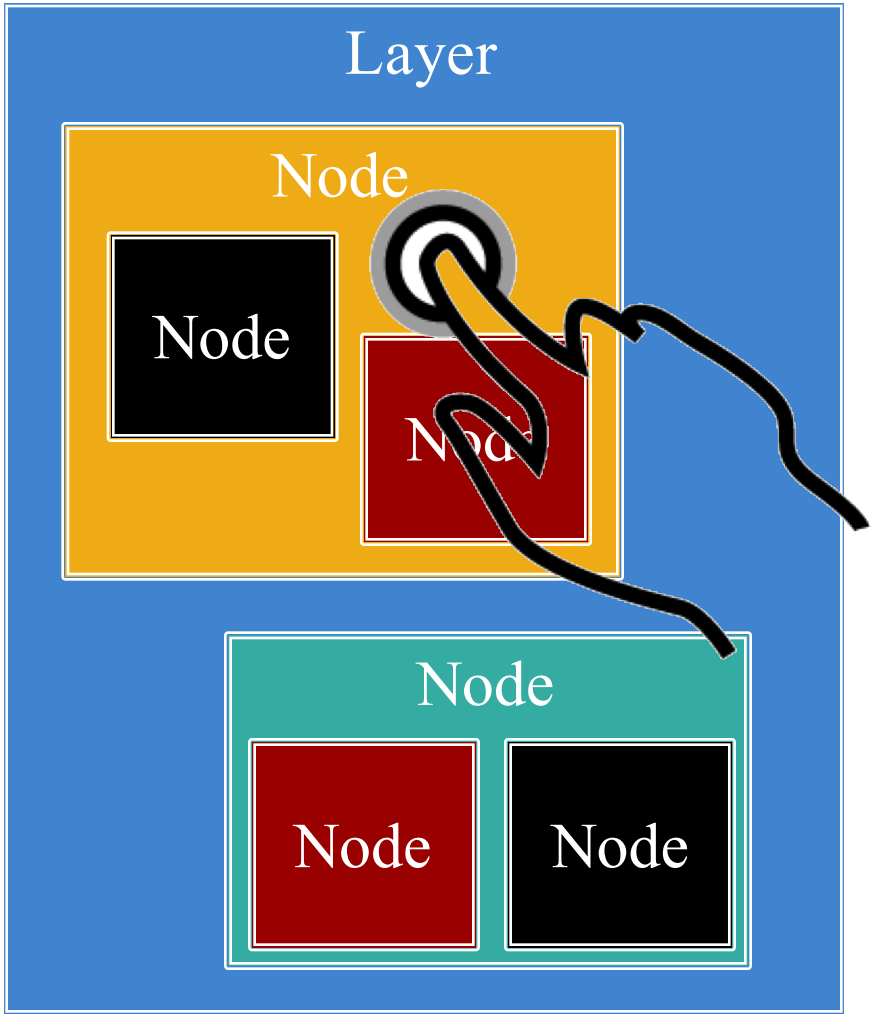
- **Animation** is much easier!
- Can reduce filmstrips
  - Break asset into parts
  - Each has a coord system
  - Transform each s
- Decouple ani
  - Update does
  - Node advances frame
  - Update switches animations
- **Exception**: ShipDemo

See Cocos2D Animation Tutorials



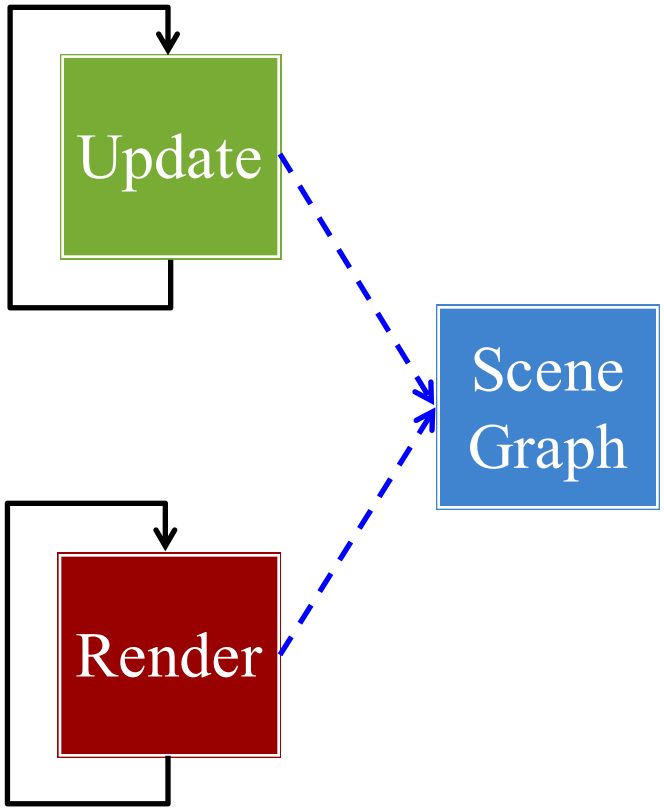
Also Good for Touch Interfaces

- Touch handler requires
  - Which object touched
  - Location inside object
- Scene graph is a *search tree*
  - Check if touch is in parent
  - … then check each child
  - Faster than linear search
- But limit this to a **search**
  - No input control in node
  - Polling over callbacks



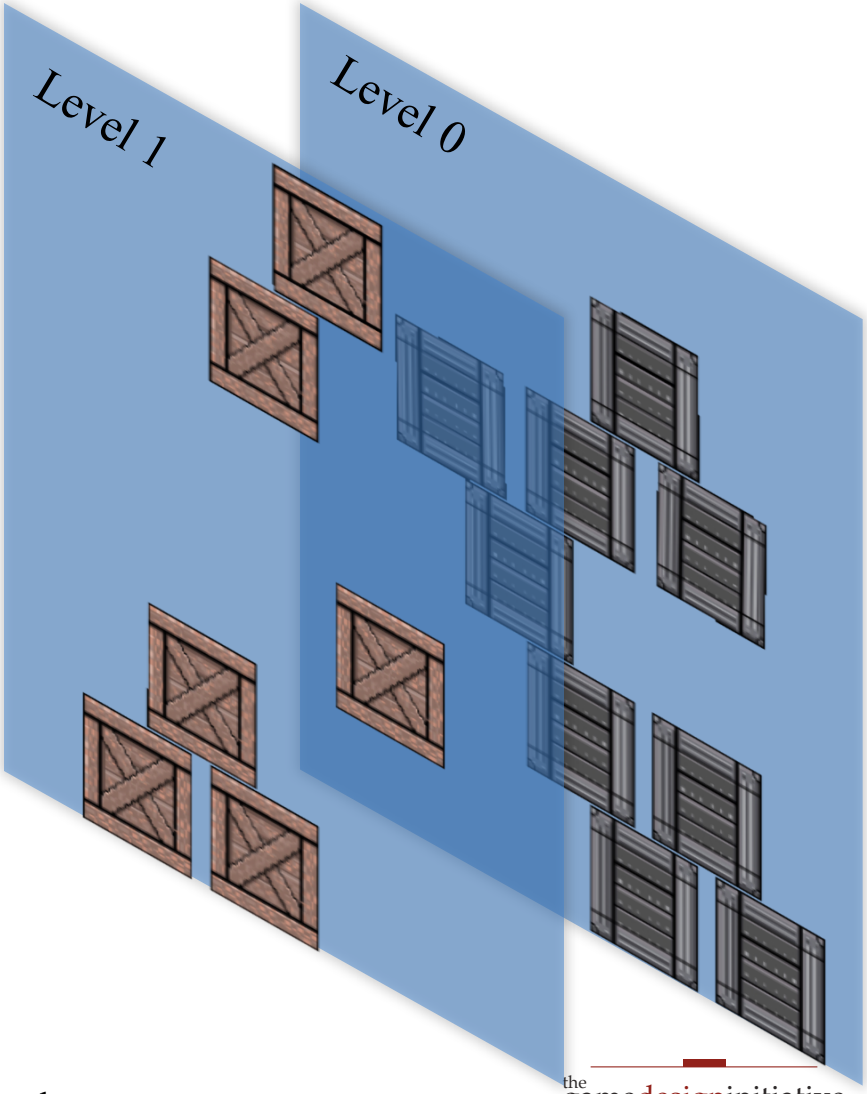
Scene Graphs + Performance?

- Decouple scene & renderer
  - Update pass modifies scene
  - Render pass to SpriteBatch
  - Draw pass sends to GPU
- Renderer in second thread?
  - Can draw even update slow
  - Decoupled animation can still look very smooth
- What Cocos2D 3.x does
  - Sort-of (**textures only**)
  - Wireframes are expensive!



Optimizing Performance: **zOrder**

- Can specify draw order
  - Give each child a *z*-value
  - Ties are permitted
  - Objects drawn by z-order (ties broken by graph order)
- Controls texture switching
  - One texture = one *z*-value
  - Reduces it to one draw call
  - **Example**: RocketDemo
- Should do when no overlap
  - Big Cocos2d optimization

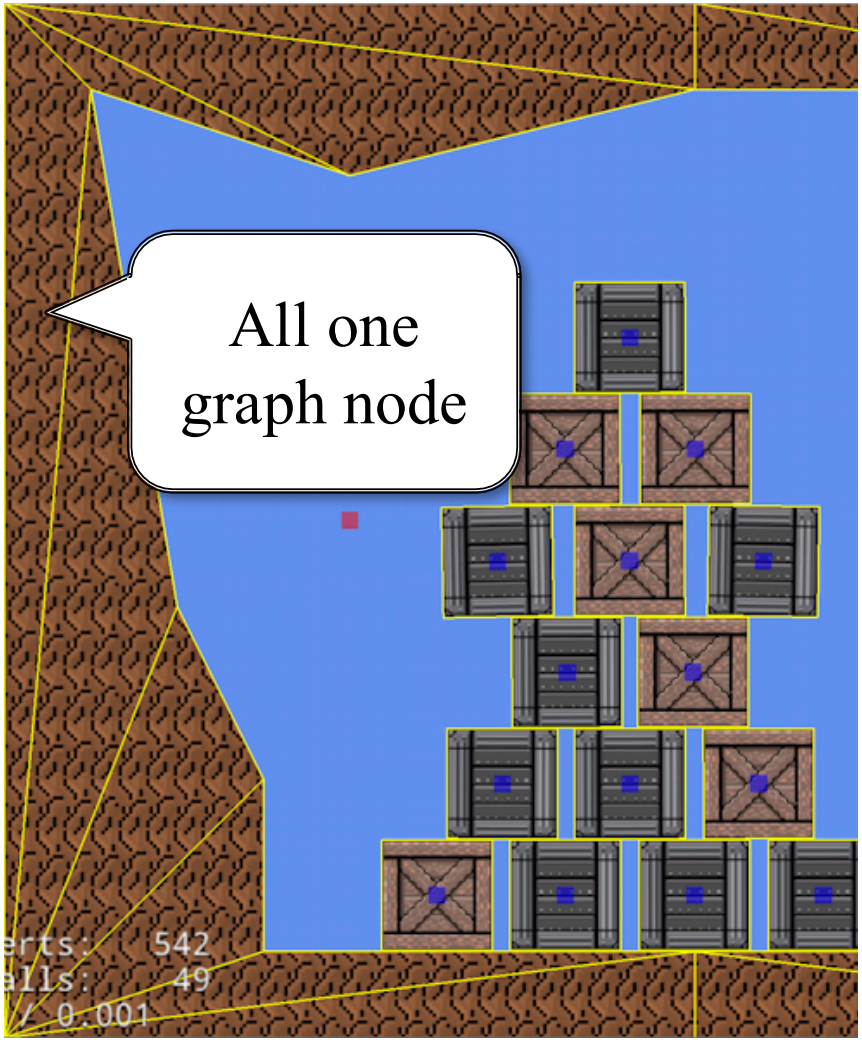


Optimizing Performance: **Atlases**

- **Idea**: Never switch textures
  - Film strip is many images
  - We can draw part of texture
  - One texture for everything?
  - Called a **texture atlas**
- Disadvantages?
  - Cannot tile textures
  - Can be tricky to pack
- Ideal for **interface design**
  - Images for UX widgets
  - Often small and compact

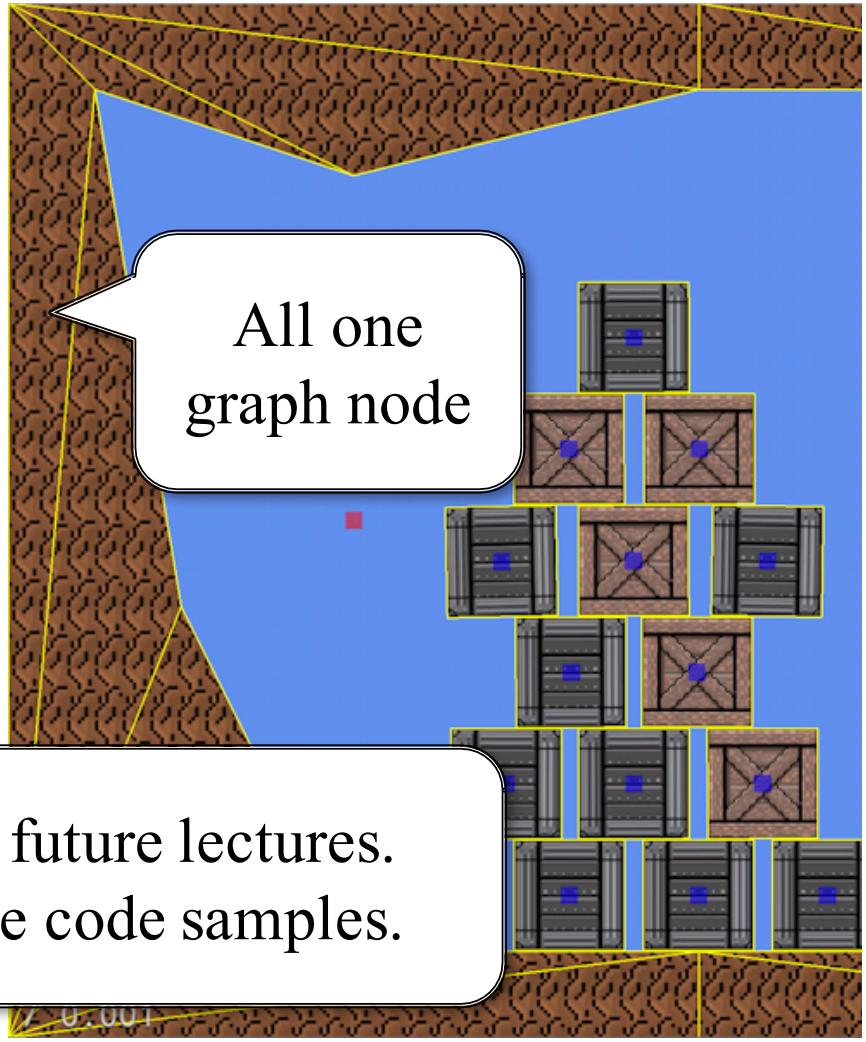
Making Custom Graph Nodes

- Demos had **custom nodes**
  - `Filmstrip` (simple animation)
  - `PolygonSprite` (tiled shapes)
- Emulate existing classes
  - `Sprite` if need textures
  - `DrawNode` if no textures
  - See how they are written
- Two main methods needed
  - Static constructor
  - The **draw** method



Making Custom Graph Nodes

- Demos had **custom nodes**
  - `Filmstrip` (simple animation)
  - `PolygonSprite` (tiled shapes)
- Emulate existing classes
  - `Sprite` if need textures
  - `DrawNode` if no textures
  - See how they are written
- Two main methods need
  - Static constructor
  - The **draw** method



In future lectures.
See code samples.

The Draw Command

```
void PolygonNode::draw(Renderer *renderer,
                       const Mat4 &transform, uint32_t flags) {

  // Don't calculate culling if transform was not updated
  _insideBounds = (flags & FLAGS_TRANSFORM_DIRTY) ?
                  renderer->checkVisibility(transform, _contentSize) :
                  _insideBounds;

  if (_insideBounds) {
    _command.init(_globalZOrder, _texture->getName(),
                  getGLProgramState(), _blendFunc, *_triangles,
                  transform);
    renderer->addCommand(&_command);
  }
}
```

The Draw Command

```
void PolygonNode::draw(Renderer *renderer,
                       const Mat4 &transform, uint32_t flags) {

  // Don't calculate [illegible]sform was not updated
  _insideBounds = ([illegible]S_TRANSFORM_DIRTY) ?
                  renderer->checkVisibility(transform, _contentSize) :
                  _insideBounds;

  if (_insideBounds) {
    _command.init(_globalZOrder, _texture->getName(),
                  getGLProgramState(), _blendFunc, *_triangles,
                  transform);
    renderer->addCommand(&_command);
  }
}
```

Check if on screen

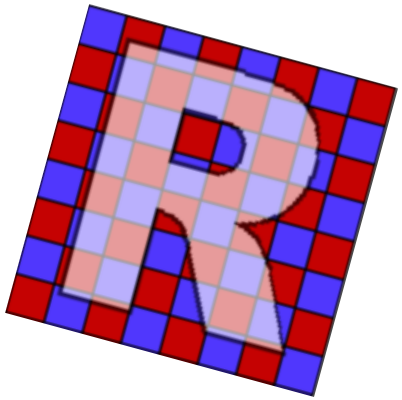
Cocos2d Class

OpenGL Parameters

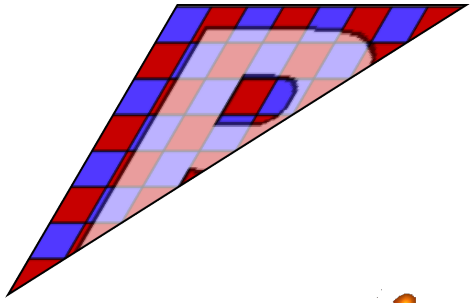
Send command to renderer

Cocos2D Command Types

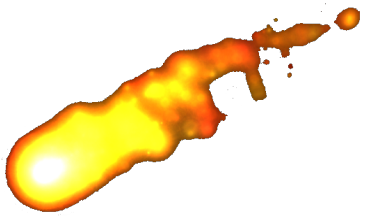
- **QuadCommand**
  - Can draw a textured rectangle
  - Batches quads together if possible
  - Used by the `Sprite`



- **TrianglesCommand**
  - Batched together, but not with quads
  - Used by `PolygonNode` (and others)



- **CustomCommand**
  - Give it a **callback function**
  - Can execute arbitrary OpenGL



Using CustomCommand

```
void DrawNode::draw(Renderer *renderer, const Mat4 &transform,
                    uint32_t flags) {

    _customCommand.init(_globalZOrder);
    _customCommand.func =
            CC_CALLBACK_0(DrawNode::onDraw, this, transform,
flags);
    renderer->addCommand(&_customCommand);
}

void DrawNode::onDraw(const Mat4 &transform, uint32_t flags){
    auto glProgram = getGLProgram();
    // OpenGL Code
    ...
}
```

Using CustomCommand

```
void DrawNode::draw(Renderer *renderer, const Mat4 &transform,
                    uint32_t flags) {
    _customCommand.init(_glob
    _customCommand.func =
        CC_CALLBACK_0(DrawNode::onDraw, this, transform,
flags);
    renderer->addCommand(&_customC
}

void DrawNode::onDraw(const Mat4 &transform, uint32_t flags){
    auto glProgram = ge
    // OpenGL Code
    ...
}
```

Method to execute

Method arguments

Method object

Draws triangle vertices created by helpers.

Using CustomCommand

```
void DrawNode::draw(Renderer *renderer, const Mat4 &transform,
                    uint32_t flags) {

    _customCommand.init(_globalZOrder);
    _customCommand.func =
            CC_CALL                                    ransform,
flags);
    renderer->add(                    _customCommand);
}

void DrawNode::onDraw(const Mat4 &transform, uint32_t flags){
    auto glProgram = get
    // OpenGL Code
    ...
}
```

You can probably subclass and just add new helpers

Draws triangle vertices created by helpers.

Summary

- Cocos2D uses **scene graphs** to draw
  - Tree of nodes with relative coordinate systems
  - Root node, `Layer`, processes all touch input
- Cocos2D has integrated **too much** into graph
  - Physics should be separate (RocketDemo)
  - Touch is useful, but separate listeners from node
- Cocos2D 3.x has made major improvements
  - Sprites and polygons are now batched together
  - But need to understand how to optimize