

Lecture 12

Data-Driven Design

Take-Away for Today

- What is “data-driven” design?
 - How do the programmers use it?
 - How to the designers/artists/musicians use it?
- What are the benefits of data-driven design?
 - To both the developer and the player
- What is a level editor and how does it work?
 - What can you do graphically?
 - How does scripting work in a level editor?

Recall: Game Components

- **Game Engine**

- Software, created primarily by programmers

- **Rules and Mechanics**

- Created by the designers, with programmer input

- **User Interface**

- Coordinated with programmer/artist/HCI specialist

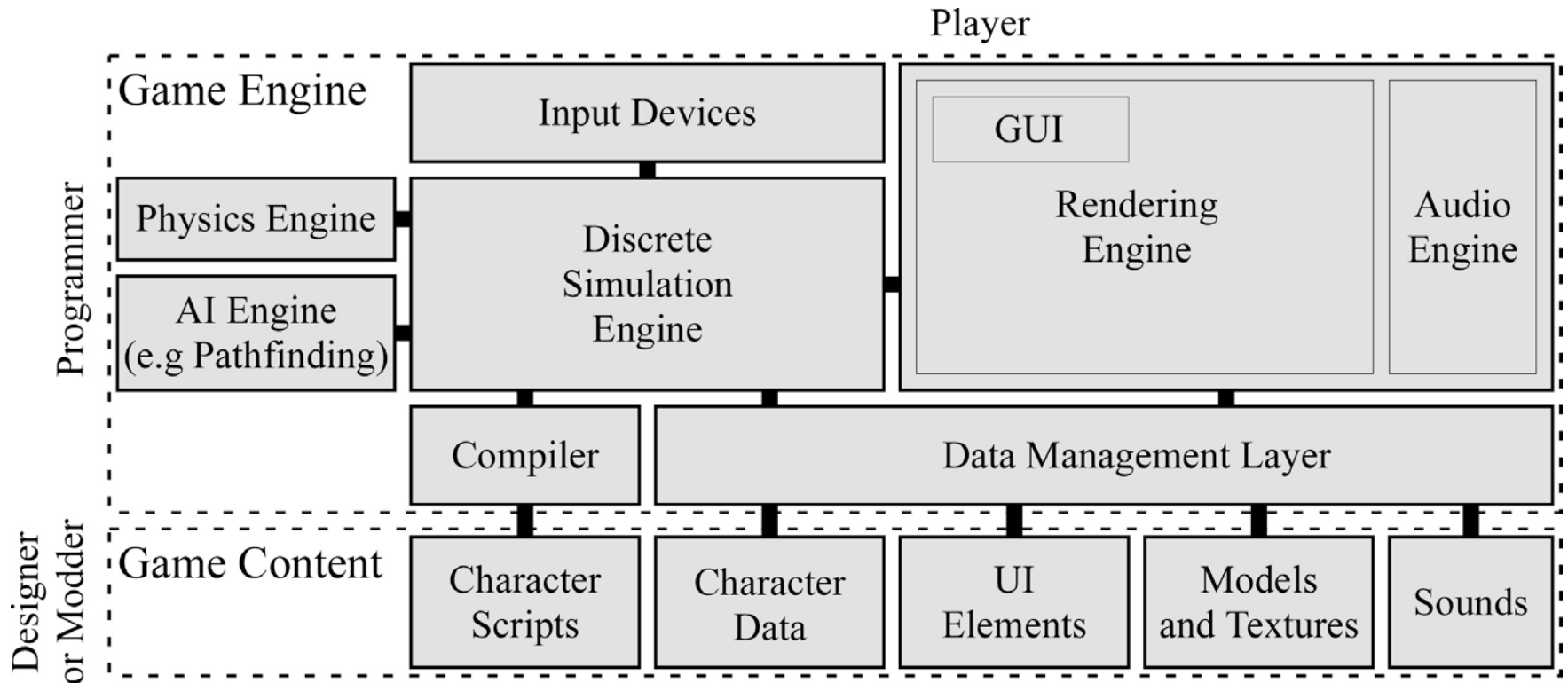
- **Content and Challenges**

- Created primarily by designers

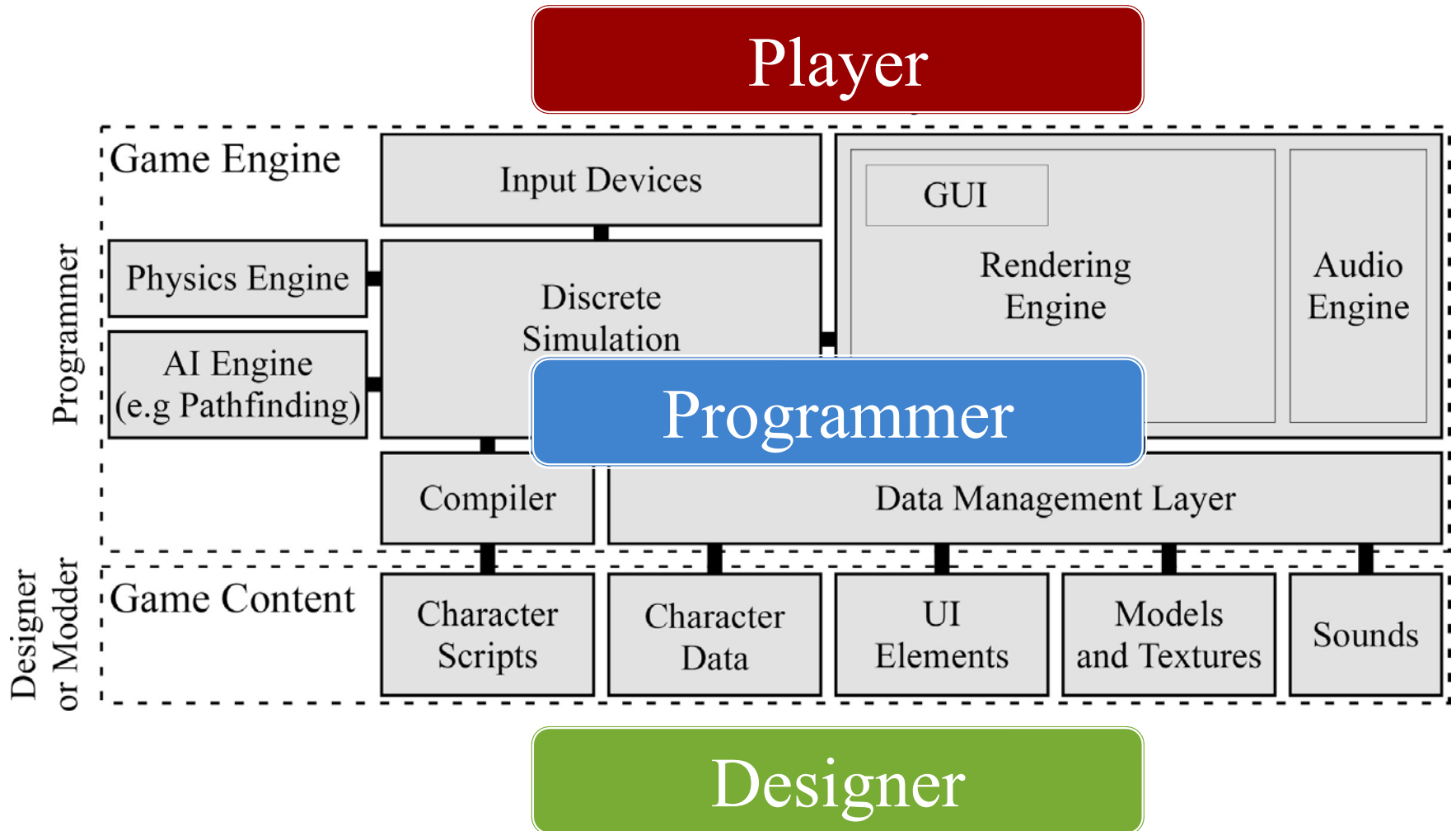
Data Driven Design

- **No code outside engine**
 - Engine determines space of possibilities
 - Actual possibilities are data/scripts
- **Examples:**
 - Art and music in industry-standard file formats
 - Object data in JSON or other data file formats
 - User interface in JSON or other data files
 - Character behavior specified through scripts

Architecture: The Big Picture



Architecture: The Big Picture



Common Development Cycle

- Start with small number of programmers
- Programmers create a **content pipeline**
 - Productivity tools for artists and designers
 - Data can be imported, viewed and playtested
- Hire to increase number of artists, designers
 - **Focus**: creating content for the game
- Ship title and repeat (e.g. cut back on artists)

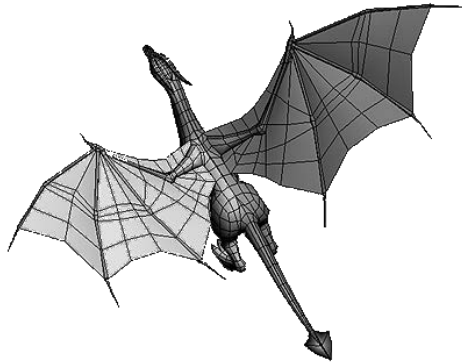
Content Pipeline

Art Tools

Initial File
Format

Final File
Format

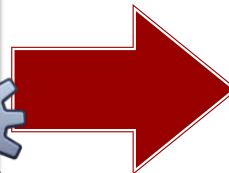
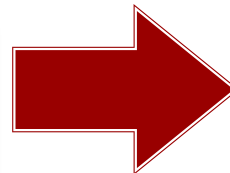
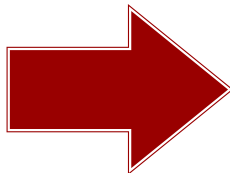
Software



AUTODESK
FBX

G3DJ

**lib
GDX**



Content Creation Tools

- **Level Editor**

- Create challenges and obstacles
- Layout the user interface
- Tune parameters (physics, difficulty, etc.)

- **Scripting Tools**

- Layout the user interface
- Define character behavior
- Script triggers and events

Level Editor Features

- **Create Terrain**

- Defines game geometry as 2D or 3D space
- Terrain can be **free-form** or as **grid tiles**

- **Place Objects**

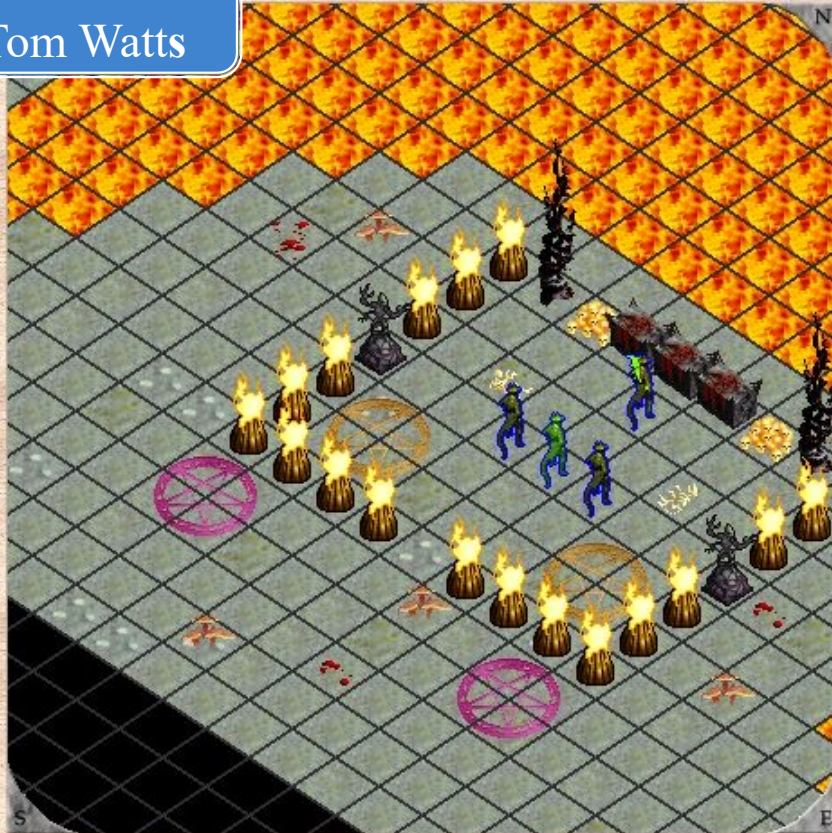
- Includes NPCs, hazards, power-ups, etc.
- Again can be free-form or aligned to a grid

- **Annotate Objects/Terrain**

- Attach scripts to interactive objects
- Define boundaries for event triggers

Example: *Blades of Avernum*

Bahssikava
by Tom Watts



Creature 68: Slith priest
Edit This Creature (Type 36, L12)
Script: Default
Attitude: Friendly
Character ID: 462

Hidden Class: 0
Drop Item 1: None
Drop Item 2: None
Personality: 0
Facing: North

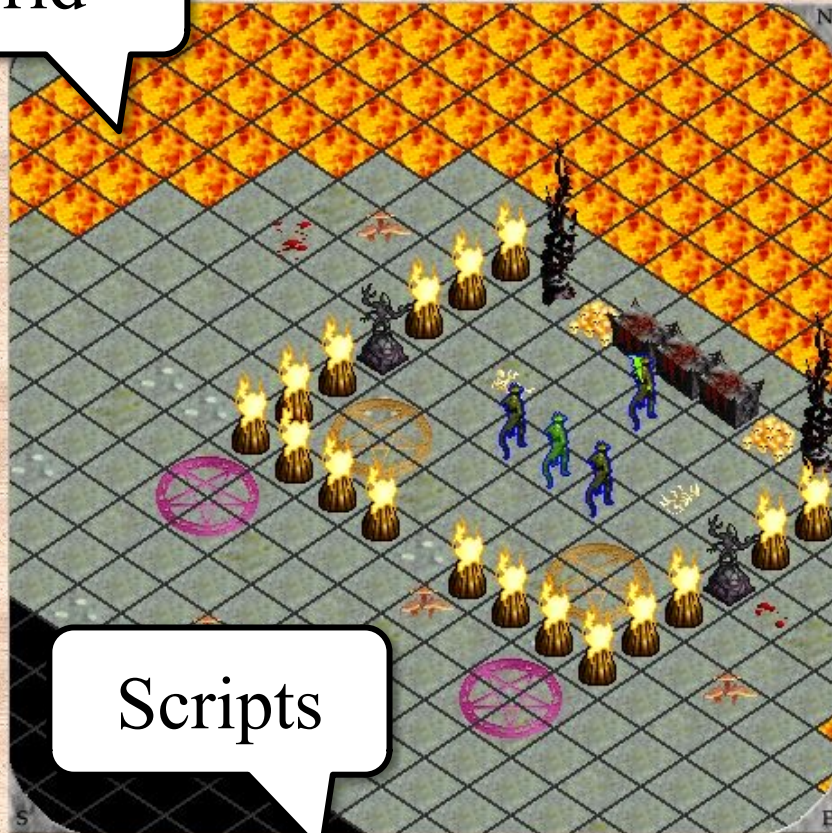


Drawing mode: FLOORS
Center: x = 32, y = 32

Select/edit placed object
Select object to edit

Example: *Blades of Avernum*

Grid



Scripts

Creature 68: Slith priest
Edit This Creature (Type 36, L12)
Script: Default
Attitude: Friendly
Character ID: 462

Hidden Class: 0
Drop Item 1: None
Drop Item 2: None
Personality: 0
Facing: North

Terrain



Tools



Level Editor: Code Sharing

- **Option:** level editor in **same project**
 - Single IntelliJ project for both
 - **Pro:** Easy to integrate into the game itself
 - **Con:** Harder to separate modules/subsystems
- **Option:** develop **core technology**
 - Identify source code used by each
 - JAR for both level editor and game
 - **Pro:** Cleaner separation of subsystems
 - **Con:** Harder to iterate the design

Level Editor: **Serialization**

Stores:
Game Model



Level Editor: **Serialization**

- Do not **duplicate** data
 - Art and music are separate files
 - Just reference by the file name
- Must **version** your file
 - As game changes, format may change
 - Version identifies the current file format
 - Want a **conversion utility** between versions
 - Version should be part of **file header**



Standard Serialization Formats

XML

```
<NPC>
  <type>Orc</type>
  <health>200</health>
  <position>
    <x>50</x>
    <y>25</y>
  </position>
</NPC>
```

JSON

```
{
  "NPC" : {
    "type" : "Orc",
    "health" : 200,
    "position" : {
      "x" : 50,
      "y" : 25
    }
  }
}
```


Standard Serialization Formats

XML

```
<NPC>  
  <type>Orc</type>  
  <health>200</health>  
  <position>  
    <x>50</x>  
    <y>25</y>  
  </position>  
</NPC>
```

XmlReader

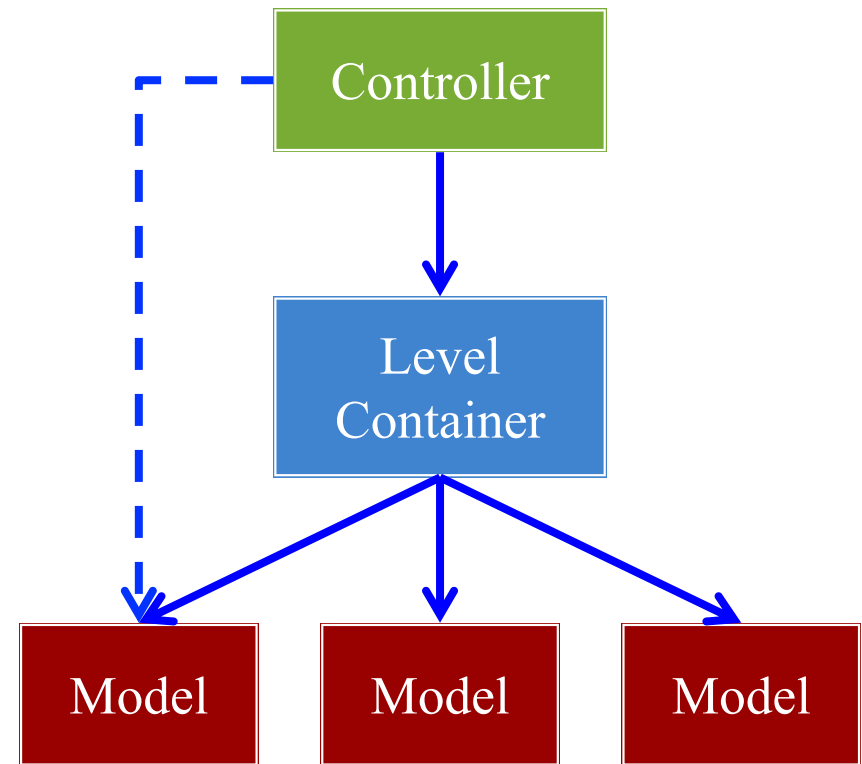
JSON

```
{  
  "NPC" : {  
    "type" : "Orc",  
    "health" : 200,  
    "position" : {  
      "x" : 50,  
      "y" : 25  
    }  
  }  
}
```

JsonReader

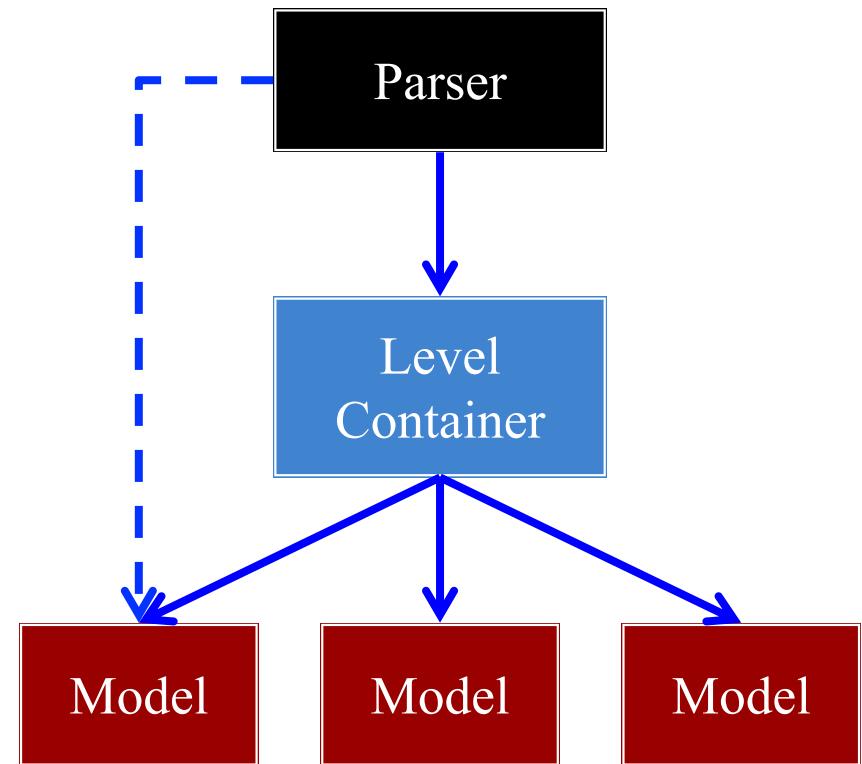
Levels and Game Architecture

- Level container (**model**)
 - Collection of model objects
 - Interface to the controllers
 - Similar to a collection type
 - May have other methods
- Level parser (**controller**)
 - Performs (de)serialization
 - Collabs with *all* models
 - Typically a factory pattern
 - Can embed *some* in model



Levels and Game Architecture

- Level container (**model**)
 - Collection of model objects
 - Interface to the controllers
 - Similar to a collection type
 - May have other methods
- Level parser (**controller**)
 - Performs (de)serialization
 - Collabs with *all* models
 - Typically a factory pattern
 - Can embed *some* in model



In-Model Deserialization

Unacceptable

```
class Model {  
  
    ...  
  
    void loadFile(String name) {...}  
  
    ...  
  
    void loadFile(File file) {...}  
  
    ...  
  
}
```

Acceptable

```
class Model {  
  
    ...  
  
    void loadData(JSON data) {...}  
  
    ...  
  
    void loadData(XML data) {...}  
  
    ...  
  
}
```

In-Model Deserialization

Unacceptable

```
class Model {  
  ...  
  void loadFile(String name) {...}  
  ...  
  void loadFile(File file) {...}  
  ...  
}
```

I/O handled
in model

Acceptable

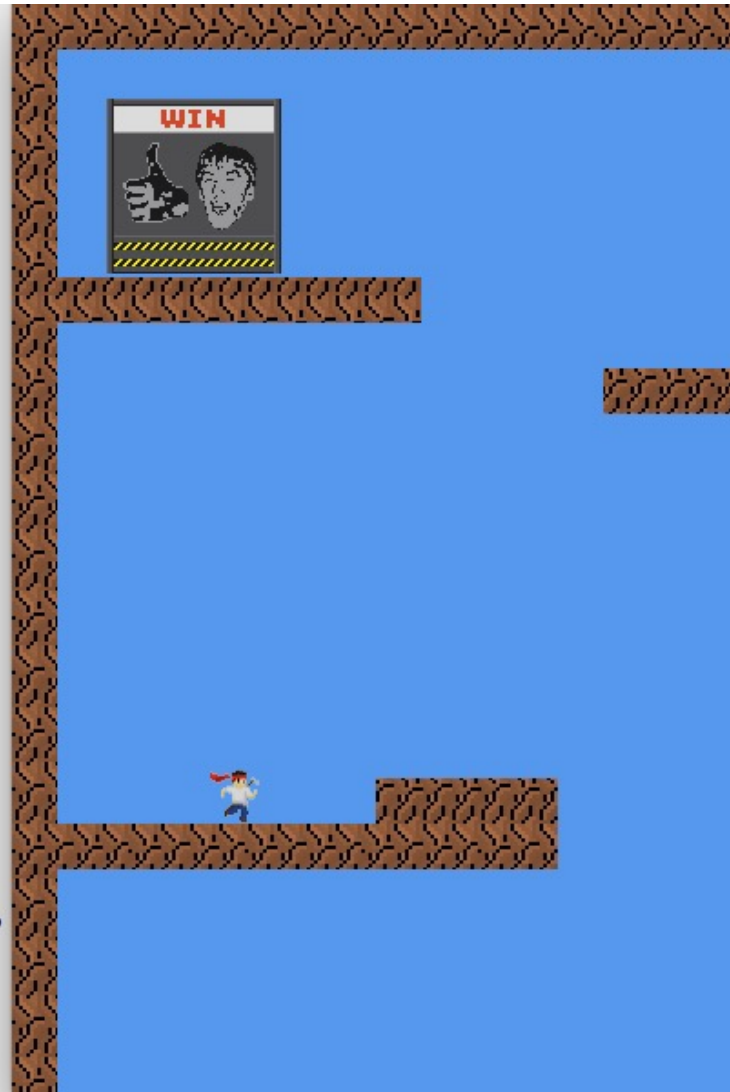
```
class Model {  
  ...  
  void loadData(JSON data) {...}  
  ...  
  void loadData(XML data) {...}
```

I/O handled
previously

I/O Code is brittle and platform-specific

Example: JSON Demo

```
"physicsSize": [16.0,12.0],
"graphicSize": [ 800, 600],
"gravity": -9.8,
"avatar": {
  "pos": [ 2.5, 5.0],
  "size": [ 0.45, 0.61],
  "bodytype": "dynamic",
  "density": 1.0,
  "friction": 0.0,
  "restitution": 0.0,
  "force": 10.0,
  "damping": 10.0,
  "maxspeed": 5.0,
  "jumppulse": 2.25,
  "jumplimit": 30,
  "jumpsound": "jump",
  "texture": "dude",
  "debugcolor": "white",
  "debugopacity": 192,
  "sensorsize": [ 0.183, 0.05 ],
  "sensorname": "dudeGroundSensor",
  "sensorcolor": "red",
  "sensoropacity": 192,
},
"exit": {
```



I/O is Brittle and Platform Specific

- Not all platforms **specify files** in the same way
 - Windows uses \ for directories, not /
 - Only Windows maps drives to letters
 - macOS is not *case sensitive* but .jar files are
- Not all platforms allow you to **read/write files**
 - macOS restricts access to Desktop/Documents
 - Application must get *permission* first
- Some platforms have **no file system** at all!
 - iOS and Android only have *application data*
 - But no concept of folders or directories

LibGDX Has Three File Types

- **Internal:** Read-Only
 - Location where the assets are stored
 - Could be inside of a .jar file!
- **Local:** Read-Write (**Preferences**)
 - Special save directory dedicated to your game
 - Maybe hard to find (Library folder on macOS)
 - But it is always guaranteed to exist
- **External:** Read-Write
 - The folder the application/.jar is located
 - You rarely have permission to write here

LibGDX Has Three File Types

- **Internal: Read-Only**

- Assets are stored
- **Assets** jar file!

See LibGDX File class for more

- **Local: Read-Write**

- Special save directory dedicated to your game
- **Saved Games** library folder on macOS)
- But it is always guaranteed to exist

- **External: Read-Write**

- **Do Not Use** ation/.jar is located
- **Do Not Use** permission to write here

Levels and Error Detection

- Game data is **not compiled** into software
 - Files go into a well-define folder
 - Game loads everything in folder at start-up
 - Adding new files to folder adds levels
- But this requires **robustness**
 - What if the levels are **missing**?
 - What if the levels are **corrupted**?
 - What if you are using **wrong file version**?



Levels and Error Detection

- **Corruption** a major problem in this design
 - Player might trash a level file (or directory)
 - Modder might alter level improperly
 - Content patch might have failed
- Process all errors **gracefully**
 - Check **everything** at load time
 - If level corrupt, allow play in others
 - Give helpful error messages



Content Creation Tools

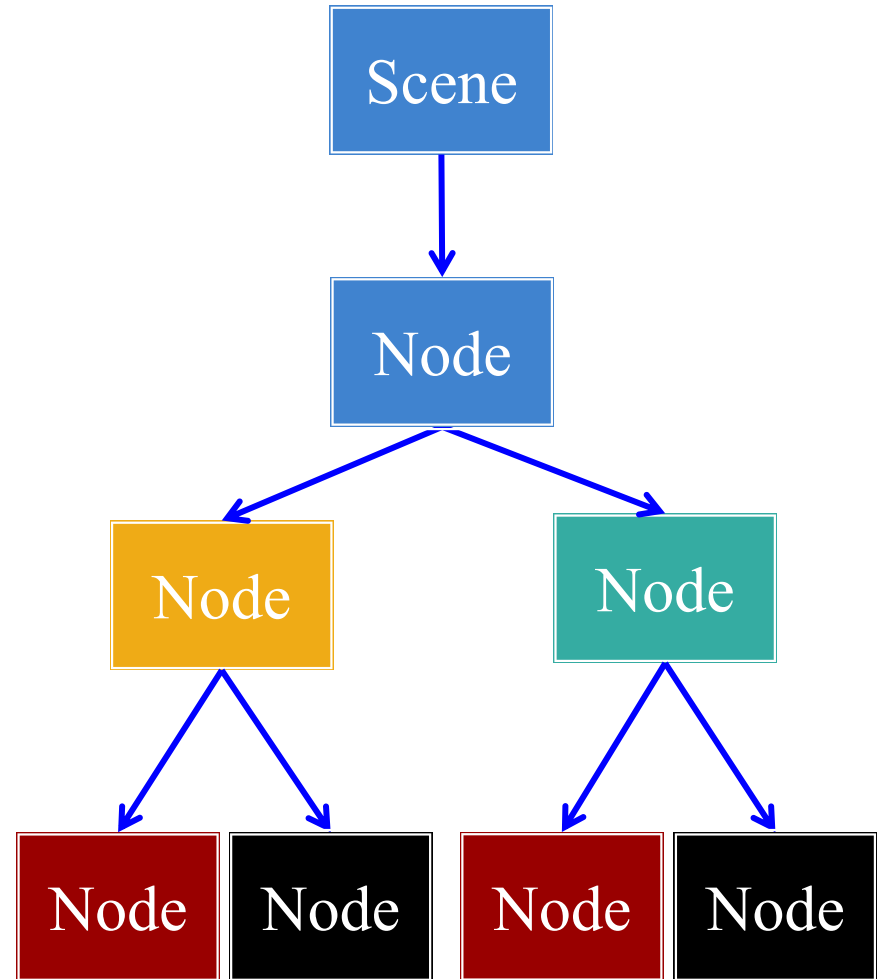
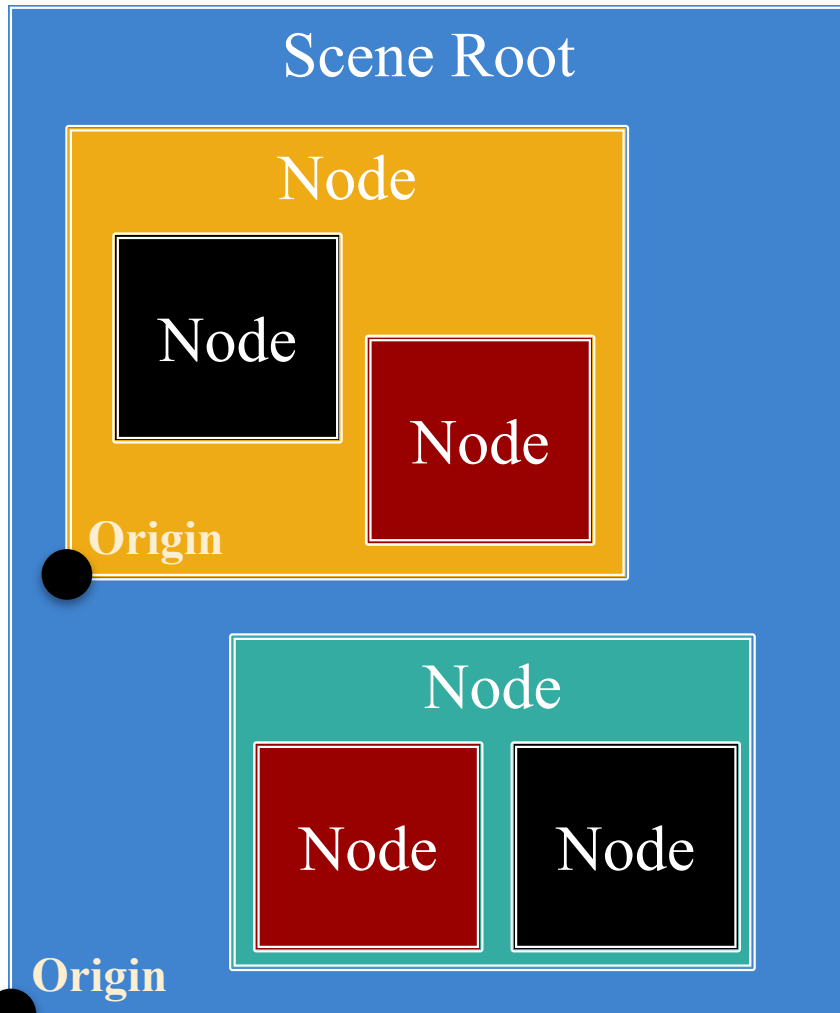
- **Level Editor**

- Create challenges and obstacles
- Layout the user interface
- Tune parameters (physics, difficulty, etc.)

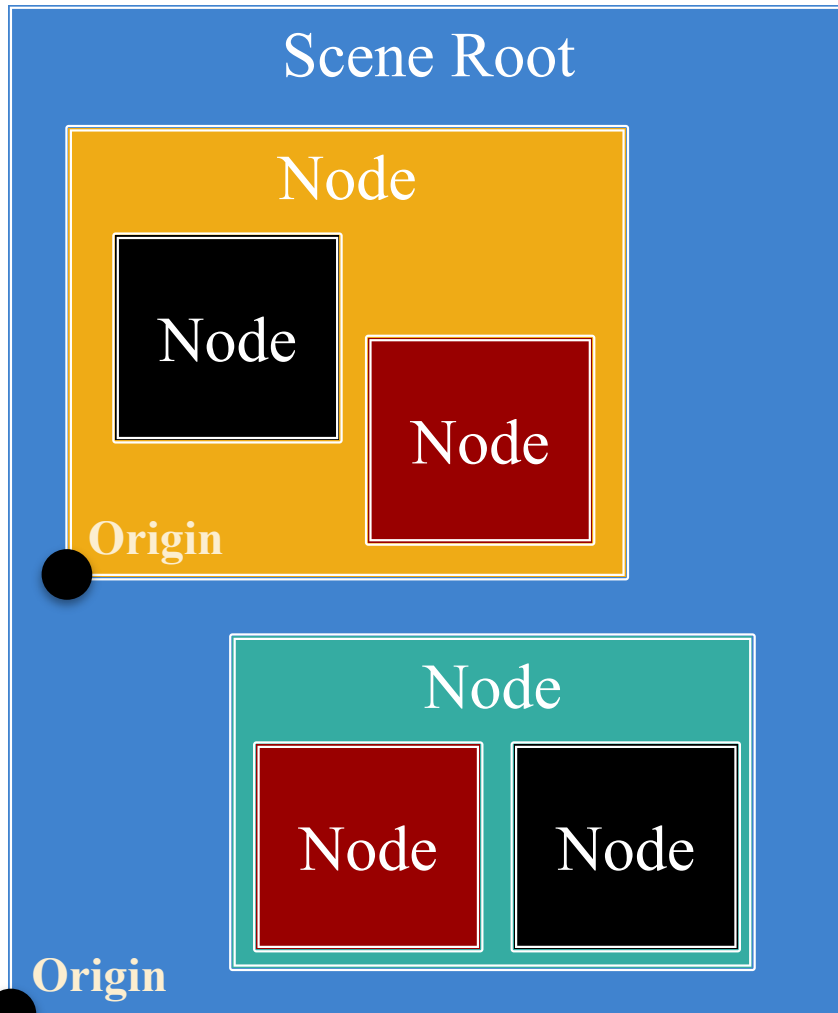
- **Scripting Tools**

- Layout the user interface
- Define character behavior
- Script triggers and events

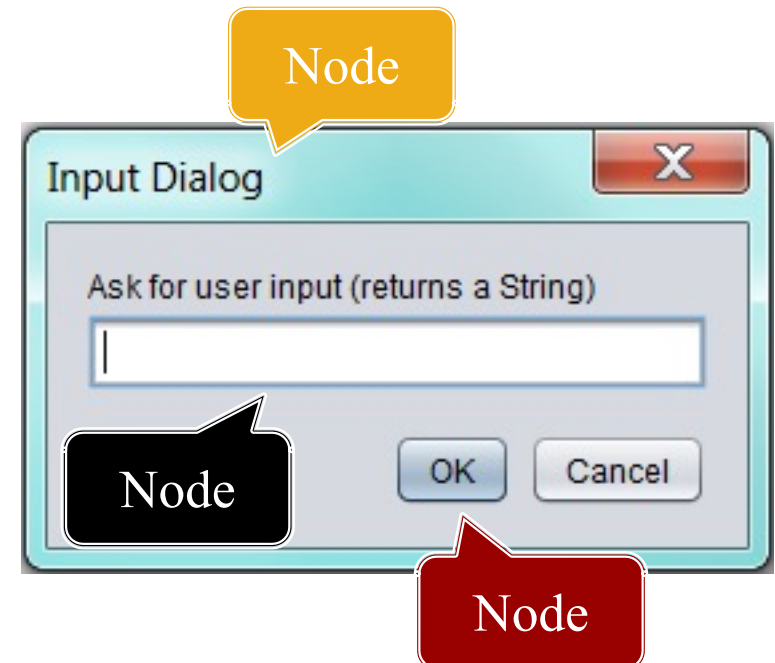
UI Design: Scene Graphs



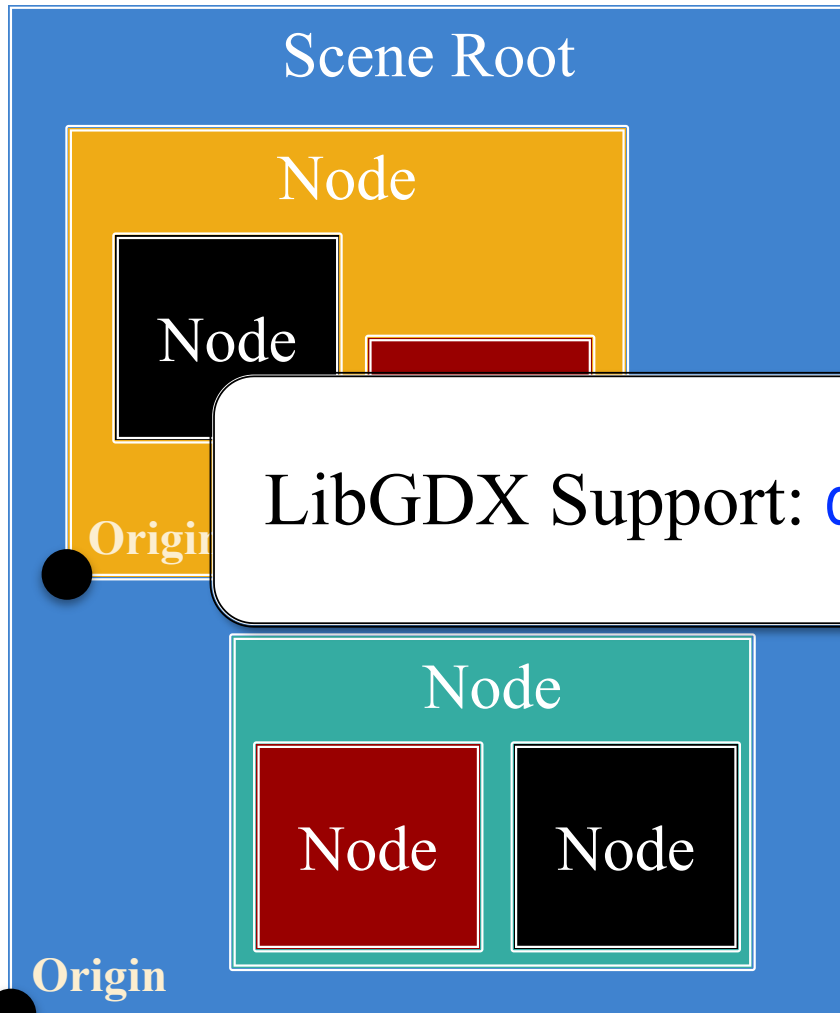
UI Design: Scene Graphs



- Node is a coordinate system
 - Logically a “window”
 - Children move with parent
- Hierarchically build widgets

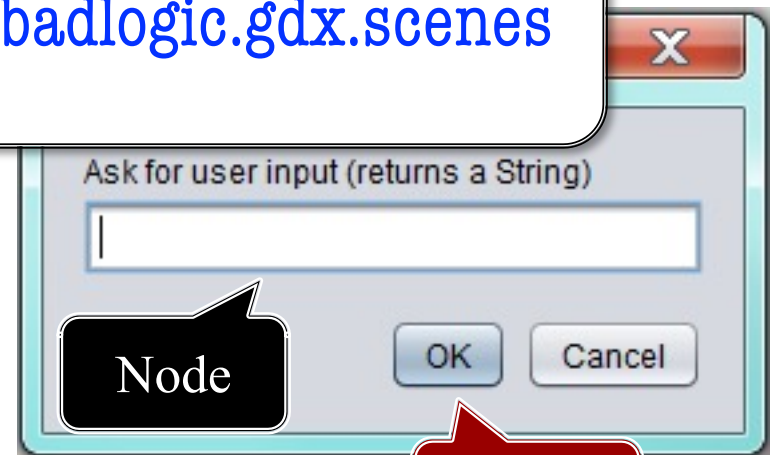


UI Design: Scene Graphs

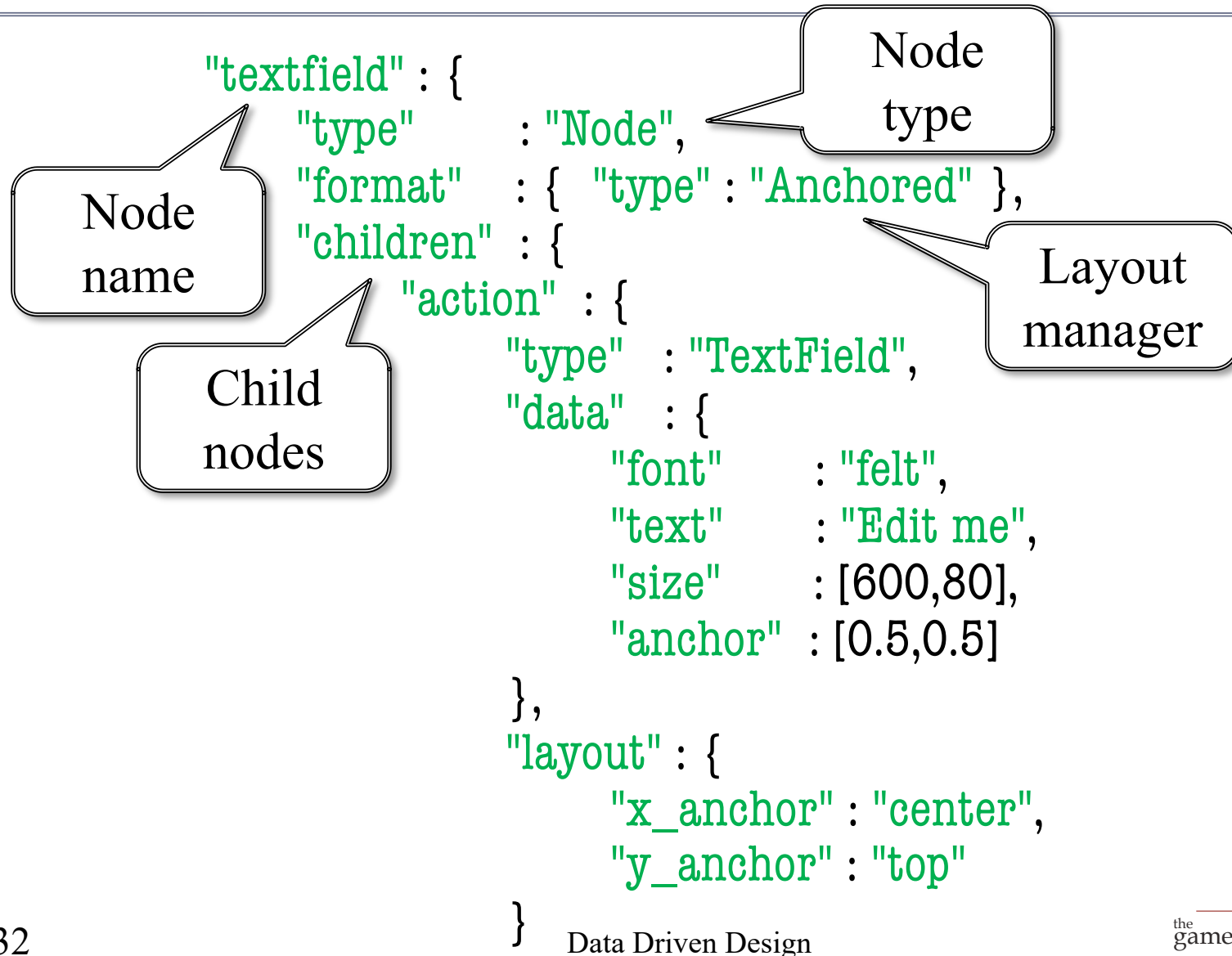


- Node is a coordinate system
 - Logically a “window”
 - Children move with parent
- Hierarchically build widgets

LibGDX Support: com.badlogic.gdx.scenes



CUGL: JSON for Scene Graphs



CUGL: JSON for Scene Graphs

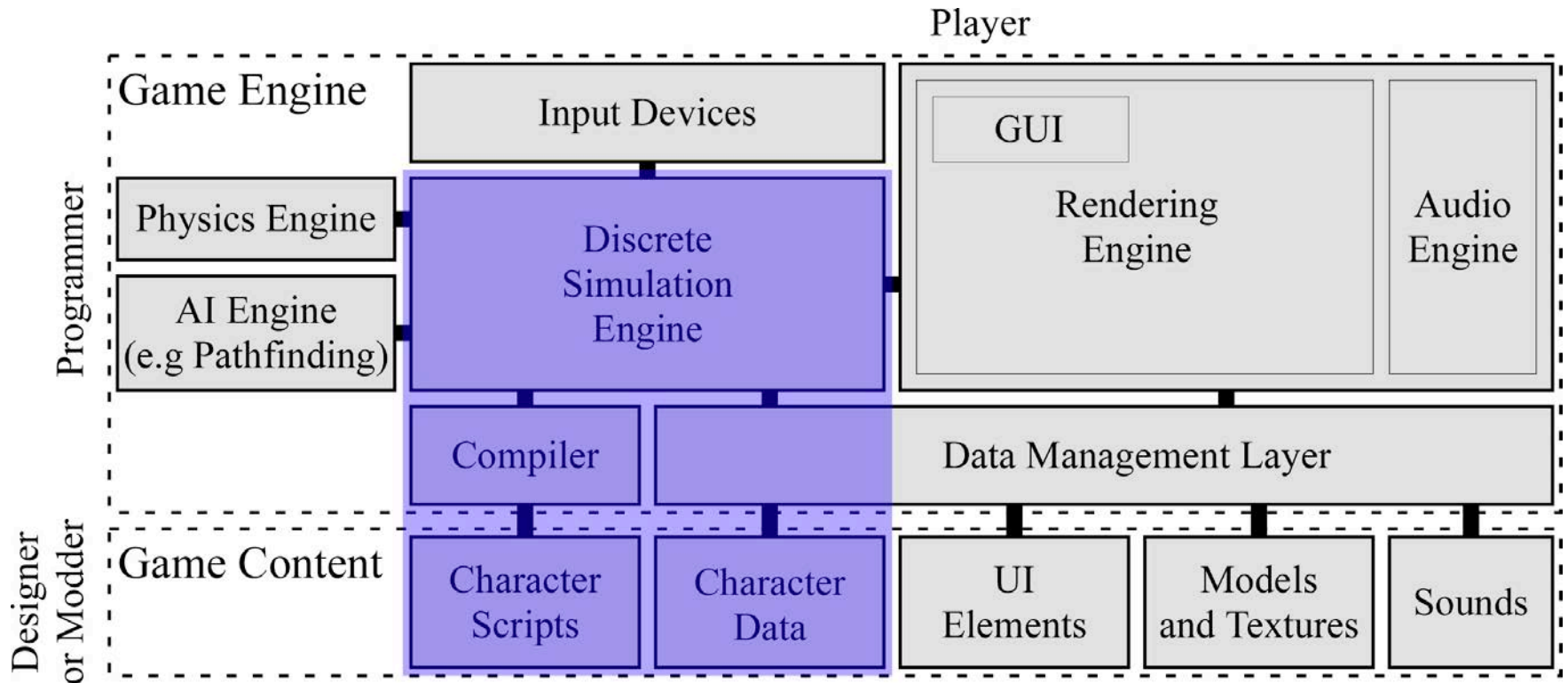
```
"textfield" : {  
  "type"      : "Node",  
  "format"    : { "type" : "Anchored" },  
  "children"  : {  
    "action" : {  
      "type" : "TextField",  
      "data" : {  
        "font"   : "felt",  
        "text"   : "Edit me",  
        "size"   : [600,80],  
        "anchor" : [0.5,0.5]  
      },  
      "layout" : {  
        "x_anchor" : "center",  
        "y_anchor" : "top"  
      }  
    }  
  }  
}
```

Layout manager

Node data

Info for parent layout

Scripting Languages



Why Scripting?

- **Character AI**

- Software only aware of high level actions
- Specific version of each action is in a script

- **Triggers**

- Actions happen in response to certain events
- Think of as an `if-then` statement
 - **if**: check if trigger should fire
 - **then**: what to do if trigger fires

Triggers and Spatial Boundaries



Launch cut scene
if Mario reaches
this box alive

Ways of Scripting

- Static **functions/constants** exposed in editor
 - Script is just the name of function to call
 - Used in the sample level editor
 - Typically good enough for this course
- Use standard **scripting language**
 - **Examples:** Lua, stackless python
 - A lot of overhead for this class
 - Only if writing high performance in C/C++

Scripting in *Dawn of War 2*

```
infantry-plan.squadai * Sc1
File Edit Search View Tools Options Language Buffers Help
1 assault-building-plan.lua 2 sniper-plan.squadai 3 infantry-plan.squadai *
16 -----
17 -- plan
18
19 plan =
20 -{
21
22 -----
23 -- phase0: Before First Bound
24 {
25     type = DATA_PHASE,
26     name = "START PLAN: all move NO COVER NO BACKWARDS",
27     --
28     {
29         apply_to = {ET_Core, ET_RFlank, ET_LFlank},
30         actions =
31         {
32             ACTION_MOVE_POSTURE_EXT( DT_MAX_SQUAD_RANGE, .95, 4.0, 30.0, "squad_formation/squad_ai.lua"
33         },
34     },
35 }
36
37 type = DATA_PHASE,
38 name = "1st SQUAD BOUND -- LOOK FOR COVER",
39 --
40 {
41     apply_to = {ET_Core, ET_RFlank, ET_LFlank},
42     actions =
43     {
44         ACTION_MOVE_POSTURE( DT_MAX_SQUAD_RANGE, 0.85, 10.0, 60.0, "squad_formation/squad_ai.lua",
45     },
46 },
47
48 -----
49 -- phase2 -- BOUND 1 CORE (core runs in an drops to prone)
50 -{
51     type = DATA_PHASE,
52     name = "Core 1st Bound"
```

Simpler: XML Specification

The screenshot shows the 'Attribute Editor' window for a game asset named 'squad_plan'. The interface is split into two main panes: a 'Tree View' on the left and a 'List View' on the right.

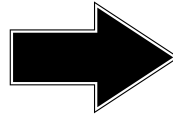
Tree View: A hierarchical tree of assets under the 'pvp' folder. The 'eld_teleport_range' asset is selected and highlighted in blue.

List View: A detailed view of the selected 'eld_teleport_range' attribute. It shows a tree structure of sub-attributes and their values:

- squad_plan_bag
 - plan_phases
 - 0. plan_phase: types\plan_phase
 - 1. plan_phase: types\plan_phase
 - 2. plan_phase: types\plan_phase
 - 3. plan_phase: types\plan_phase
 - debug_phase_name: ----- Core bound 1st BOUND -----
 - phase_finished_mode: ranged_combat
 - plan_actions
 - 0. plan_action_entry: types\plan_action_entry
 - plan_action: types\plan_actions\maleable_move
 - move_info: types\pathfinding\move_info
 - allow_backwards_move: True
 - allow_leave_los: False
 - always_move: False
 - always_move_if_not_in_cover: True
 - always_move_if_not_in_max_range: True
 - chance_to_jump: 1
 - cover_search_angle: 360
 - cover_search_radius: 8
 - face_target_after_move: True
 - formation: formation\eldarline
 - max_order_delay_secs: 2
 - min_dist_from_target: 10
 - min_order_delay_secs: 1
 - move_distance_percentage: 0.75
 - move_distance_type: min_squad_range
 - speed_multiplier_max: 1.5
 - speed_multiplier_min: 1.5

JSON/XML as a “Scripting Language”

```
"myevent" : {  
  "id" : 4,  
  "sparkle" : {  
    "color" : "blue",  
    "size" : 2,  
    "duration" : 3,  
  },  
  "buff" : {  
    "attrib" : "health",  
    "value" : 4,  
  },  
  "sound" : "magic4"  
}
```

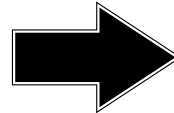


```
codefrag = "  
switch (triggerId) {  
  ...  
  case 4:  
    sparkleCharacter(BLUE,2,3);  
    buffCharacter(HEALTH,4);  
    playSound(MAGIC4);  
    break;  
  ...  
}"
```

This is text, not
compiled code

JSON/XML as a “Scripting Language”

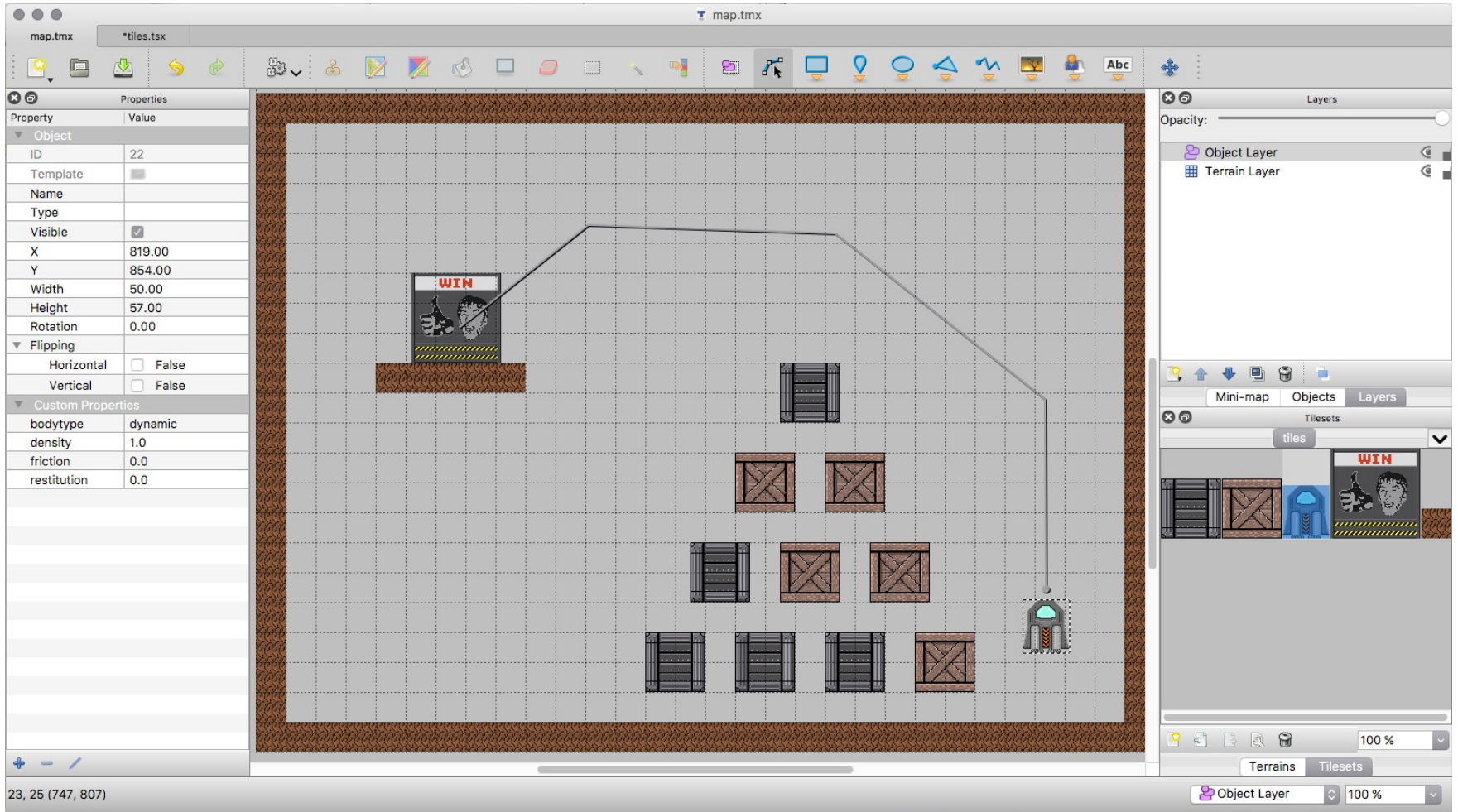
```
codefrag = "  
switch (triggerId) {  
...  
case 4:  
    sparkleCharacter(BLUE,2,3);  
    buffCharacter(HEALTH,4);  
    playSound(MAGIC4);  
    break;  
...  
}"
```



```
class MyEvent implements Event {  
    void process(int triggerId) {  
        switch (triggerId) {  
            ...  
            case 4:  
                sparkleCharacter(BLUE,2,3);  
                buffCharacter(HEALTH,4);  
                playSound(MAGIC4);  
                break;  
        }  
    }  
}
```

Java Support: `javax.tools.JavaCompiler`

Final Words: The Tiled Level Editor



Using Tiled for 3152

Advantages

- Supports **almost any game**
 - Only places terrain/objects
 - You interpret placement
 - Allows custom properties
- Supports **custom collisions**
 - Each object has a “hit box”
 - Not just rectangular shapes
- Supports **XML and JSON**

Disadvantages

- No **polygonal terrain**
 - Terrain must fit to the grid
 - NOT how Lab 4 works
- No (real) **AI scripting**
 - At best have “JSON scripts”
 - Also can define patrol paths
- No **built-in parser**
 - To convert JSON to classes

No Built-in Parser?

The screenshot shows a web browser window with the URL `libgdx.badlogicgames.com/nightlies/docs/api`. The page displays the API documentation for the `TiledMapRenderer` class. The left sidebar shows a navigation tree with `com.badlogic.gdx.maps.tiled` selected. The main content area shows the `TiledMapRenderer` interface, including its superinterfaces (`MapRenderer`), implementing classes, and a method summary table.

Method Summary

All Methods	Instance Methods	Abstract Methods
	Modifier and Type	Method and Description
	void	<code>renderImageLayer(TiledMapImageLayer layer)</code>
	void	<code>renderObject(MapObject object)</code>
	void	<code>renderObjects(MapLayer layer)</code>
	void	<code>renderTileLayer(TiledMapTileLayer layer)</code>

Methods inherited from interface `com.badlogic.gdx.maps.MapRenderer`

`render, render, setView, setView`

No Built-in Parser?

The image shows a screenshot of a web browser displaying the libgdx API documentation for the `TiledMapRenderer` class. The browser's address bar shows the URL `libgdx.badlogicgames.com/nightlies/docs/api`. The page title is `TiledMapRenderer (libgdx API)`. The navigation menu includes `OVERVIEW`, `PACKAGE`, `CLASS` (selected), `USE TREE`, `DEPRECATED`, `INDEX`, and `HELP`. The left sidebar shows a tree of classes, with `com.badlogic.gdx.maps.tiled.renderers` highlighted in a red box. The main content area shows the `Interface TiledMapRenderer` with its superinterfaces (`MapRenderer`) and implementing classes (`BatchTiledMapRenderer`, `OrthoCachedTiledMapRenderer`). A large red banner with the text **Forbidden!** is overlaid on the page. The bottom of the browser shows the open URL: `https://libgdx.badlogicgames.com/nightlies/docs/api/com.badlogic.gdx.maps.tiled.renderers/OrthoCachedTiledMapRenderer.html`.

The Problem with External Editors

- Editors often come with **runtimes**
 - Premade classes for the editor objects
 - Parser converts JSON/XML into these classes
- This shackles your architecture design
 - You must design your classes around these
 - They often violate MVC in hideous ways
- Reject tools that screw up your architecture!
 - Good tools should be *decoupled* (e.g. Box2d)

Summary

- Data-driven design has several advantages
 - Faster content production; code reuse is easier
 - Embrace of modder community can add value
- Two major focuses in data-driven design
 - **Level editors** place content and challenges
 - **Scripts** specify code-like behavior outside of code
- Be careful with 3rd party editors
 - Can streamline your development process
 - But it can also screw up your architecture