CS354R

DR SARAH ABRAHAM

GUI AND SOUND

GRAPHICAL USER INTERFACES

WHAT IS IN A GUI?

- Not just art assets!
- GUIs display important information for the player:
  - Character status
  - Enemy status
  - Leveling information
  - Map information
  - Out of game menus

DESIGNING A GUI

- GUI layouts should be:
  - Intuitive to navigate
  - Intuitive to understand
  - Intuitive to access
- This is harder than it sounds
- An entire area of design is dedicated to interaction
- You will probably get it wrong the first time
- Iterate GUI design via user testing

GUI TYPES: MENUS

- Outside of game play options, modes, and information

Metal Gear Solid V

GUI TYPES: HUDS (HEADS UP DISPLAYS)

- In-game persistent display of information

Final Fantasy XIV

GUI TYPES: DIEGETIC DISPLAYS

- In-game display of information incorporated into world

Dead Space

GUI TYPES: GUI-LESS

- No in-game display of information – purely contextual

Last Guardian

HOW ARE GUIS BUILT?

- What are things a GUI must do?
- What are things GUIs have?
- How should code be structured to support these things?

PROGRAMMING A GUI

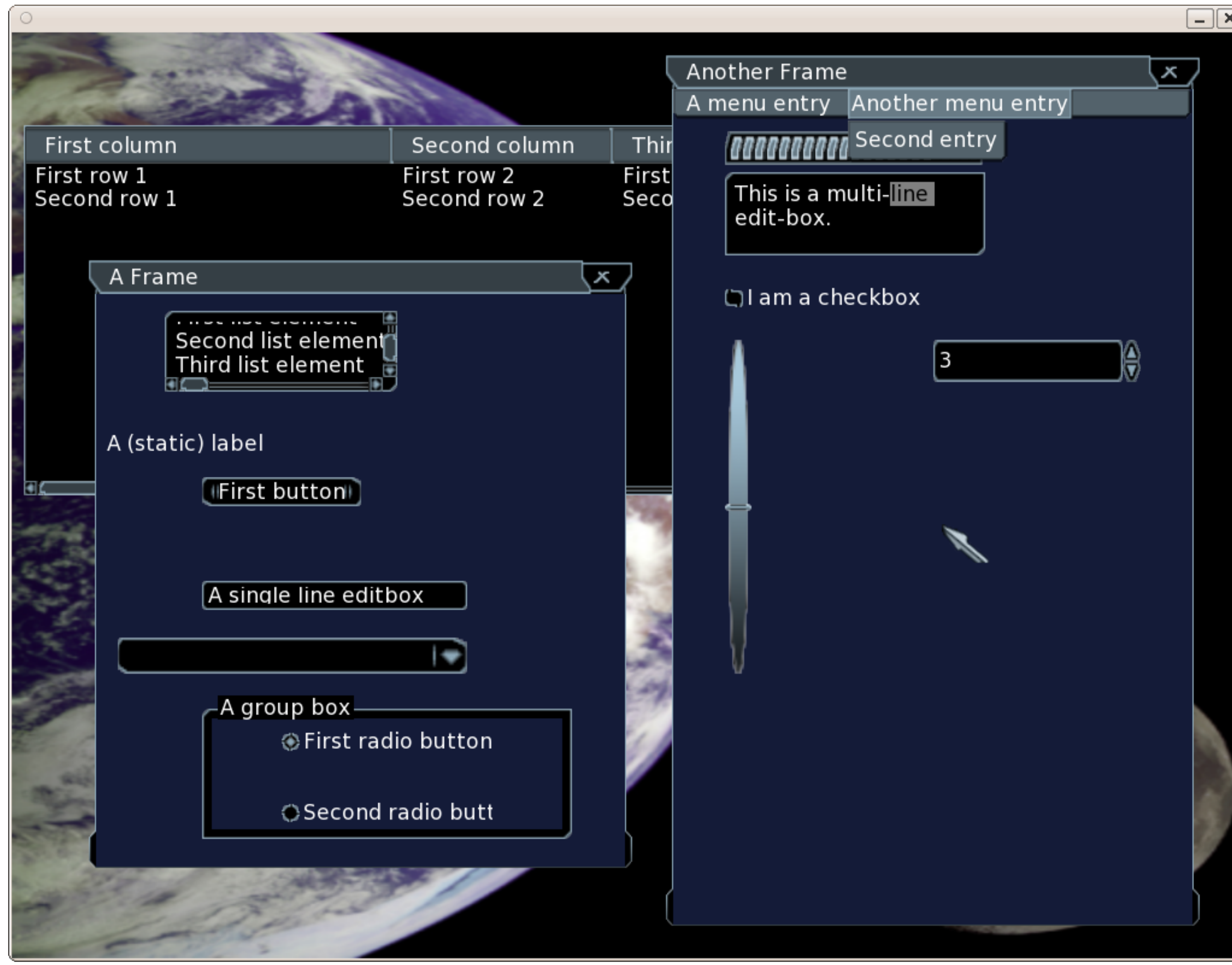
- GUI programming involves:
  - Adding widgets
  - Widget layout
  - Widget appearance
  - Widget functionality

WIDGETS

- Most GUIs are inheritance-based
- All widgets inherit from a common object type

EXAMPLE: CEGUI WINDOWS

- FrameWindow
- PushButton
- Slider
- RadioButton/Checkbox
- Editbox
- Listbox
- Menubar/PopupMenu
- Labels

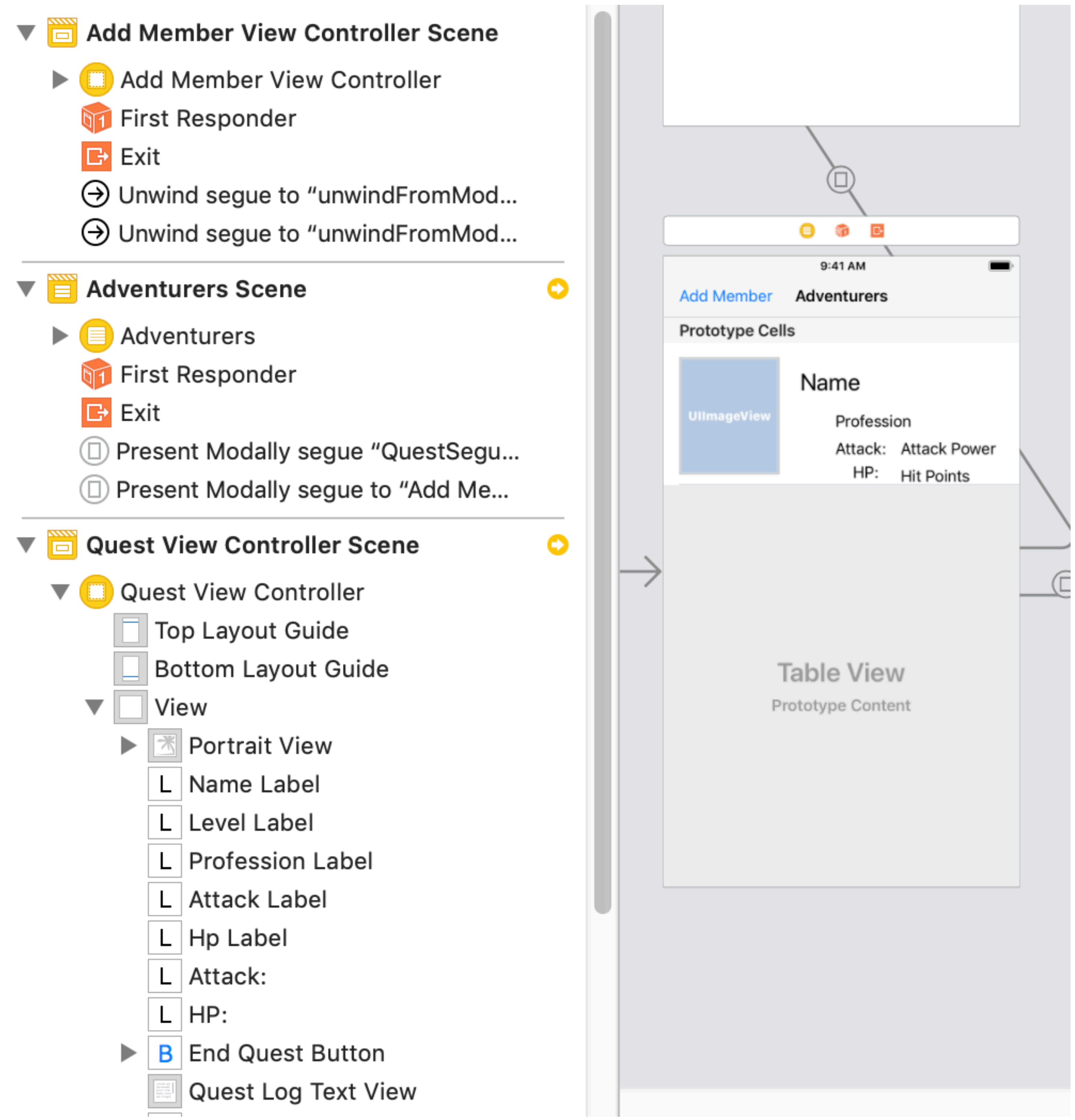


CEGUI WINDOW LAYOUT

- Parent-child relationship between container widget (window) and nested widgets
- Child settings/status inherited from parent window
  - Alpha
  - Enabled/Disabled
  - Destruction
- Window creation can be done via:
  - C++ programming
  - XML hard-coding
  - WYSIWYG (what you see is what you get) unified editor (CEED)

WYSIWYG EXAMPLE: INTERFACE BUILDER

- GUI design system built into XCode for Cocoa applications
- Auto Layout constraint system allows for resolution-independent layouts
- Outlets connect widgets to code
- Widgets use delegates to communicate with view controller (MVC model)



INTERFACE BUILDER SOURCE

- Just a giant XML document…


```xml
                </state>
                <connections>
                    <action selector="endQuestWithSender:" destination="17y-AY-3wg" eventType="touchUpInside"
                        id="B8S-VK-oAb"/>
                    <segue destination="48C-06-d6f" kind="unwind" unwindAction="unwindFromModalViewWithSender:"
                        id="Mwl-kt-1oe"/>
                </connections>
            </button>
            <textView clipsSubviews="YES" multipleTouchEnabled="YES" contentMode="scaleToFill" editable="NO"
                textAlignment="natural" translatesAutoresizingMaskIntoConstraints="NO" id="tW4-Gh-XdH">
                <rect key="frame" x="16" y="224" width="343" height="265"/>
                <color key="backgroundColor" red="1" green="1" blue="1" alpha="1" colorSpace="custom"
                    customColorSpace="sRGB"/>
                <string key="text">Lorem ipsum dolor sit er elit lamet, consectetaur cillium adipisicing pecu, sed
                    do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis
                    nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure
                    dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur
                    sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est
                    laborum. Nam liber te conscient to factor tum poen legum odioque civiuda.</string>
                <fontDescription key="fontDescription" type="system" pointSize="14"/>
                <textInputTraits key="textInputTraits" autocapitalizationType="sentences"/>
            </textView>
            <label opaque="NO" userInteractionEnabled="NO" contentMode="left" horizontalHuggingPriority="251"
                verticalHuggingPriority="251" text="Quest Log" textAlignment="natural"
                lineBreakMode="tailTruncation" baselineAdjustment="alignBaselines" adjustsFontSizeToFit="NO"
                translatesAutoresizingMaskIntoConstraints="NO" id="7DW-62-5uk">
                <rect key="frame" x="16" y="195" width="343" height="21"/>
                <fontDescription key="fontDescription" type="system" pointSize="17"/>
                <color key="textColor" red="0.0" green="0.0" blue="0.0" alpha="1" colorSpace="custom"
                    customColorSpace="sRGB"/>
                <nil key="highlightedColor"/>
            </label>
        </subviews>
        <color key="backgroundColor" red="0.66666668653488159" green="0.66666668653488159"
            blue="0.66666668653488159" alpha="1" colorSpace="custom" customColorSpace="sRGB"/>
        <constraints>
            <constraint firstItem="7DW-62-5uk" firstAttribute="top" secondItem="GYY-YM-08B" secondAttribute="bottom"
                constant="17" id="0CG-jm-fwu"/>
            <constraint firstItem="rsr-Lb-ZQg" firstAttribute="top" secondItem="MkR-PF-Obu" secondAttribute="bottom"
                constant="7" id="33P-cy-JmW"/>
            <constraint firstItem="Ob8-vZ-Sfd" firstAttribute="top" secondItem="amJ-97-kYm" secondAttribute="bottom"
                constant="8" id="61O-1J-cXn"/>
            <constraint firstItem="Ob8-vZ-Sfd" firstAttribute="leading" secondItem="rsr-Lb-ZQg"
```

XML STORAGE

- Note that XML representation isn't limited to GUIs
- Fast, modular way to load assets and properties
- Gives designers handles into the system
  - Easier to version control than binary representations but less optimized
- Avoids hard-coding objects into the codebase
- Use of WYSIWYGs or scripts avoids hard-coding XML files

GUIS IN UNREAL

- Slate is Unreal's custom UI **programming framework**
  - Unreal editor is built in Slate
  - Written in C++
  - Can customize editor panels or be used in-game
  - Primarily used for tools-building
- UMG (Unreal Motion Graphics) is Unreal's **visual UI authoring tool**
  - Built using Widget Blueprints
  - Blueprint includes layout mode and event graph mode for reacting to inputs

HOW CAN WE BE RESOLUTION INDEPENDENT?

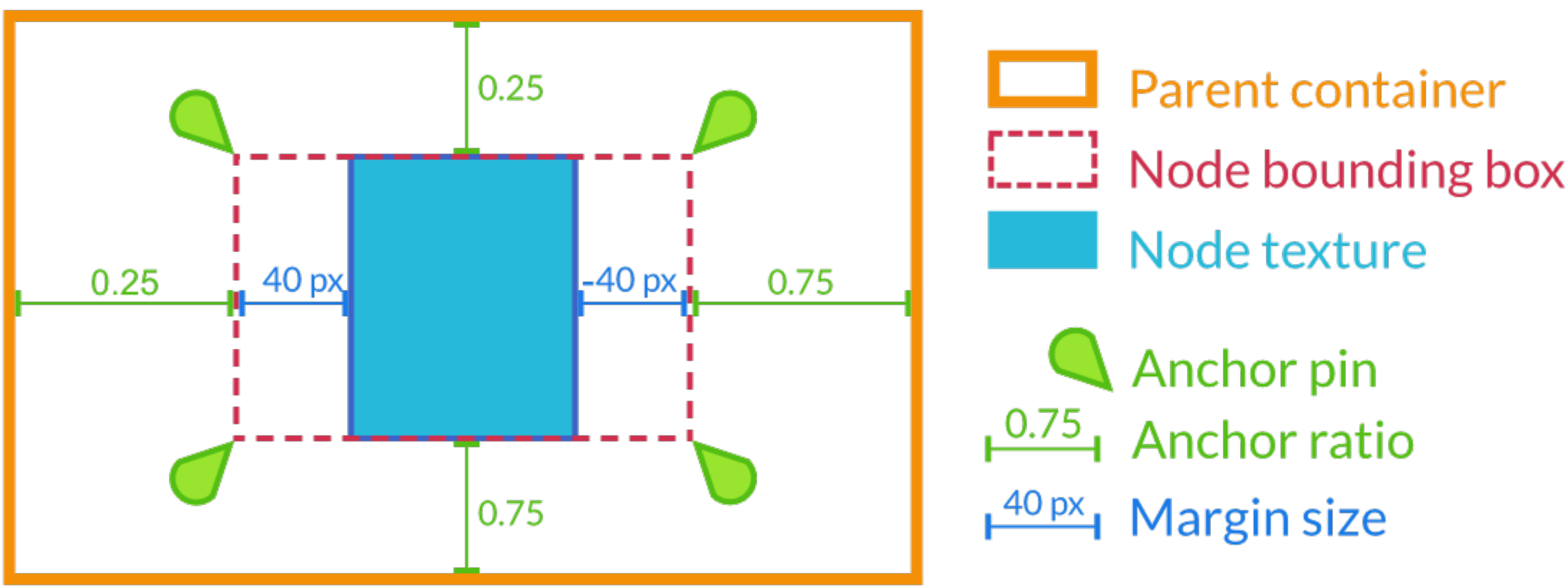
- Resolve widget placement using **constraints**
- Layout can be treated as a system of linear equations and constraints
  - Treat as an optimization problem (minimize constraint violations)
  - Resolve using a linear objective function
- Soft constraints (i.e. requested constraints that can be violated if necessary to find a solution) can be violated in non-uniform ways
  - Quadratic objective functions handle error minimization better
- Constraint solving can decrease responsiveness
- Constraint solving allows for static analysis of violations

GODOT AND GUIS

- Godot uses Control nodes for GUIs
  - Have properties for anchor points, bounding rectangle, focus, size, margin, and UI theme
- Control nodes include UI elements such as Labels, TextureRects, and TextureButtons
- TextureButtons can change state based on user interactions
  - Normal, Pressed, Hover, Disabled, Focused
- Containers are Control nodes that hold other Control nodes
  - VBoxContainer/HBoxContainer, GridContainer, CenterContainer, etc

GODOT ANCHORS

- Anchors define relative position (left, right, top, bottom) to parent container
  - Do this through the UI!



CONNECTING SCENES IN GODOT

- Godot connects GUI scenes to in-game objects using signals
  - In-game object emits a signal based on some change property (`emit_signal`)
  - GUI listens for signal and calls associated function (`connect`)
  - Both objects must exist before connection is made (`_ready` called after all nodes loaded)
- Can traverse the whole scene graph (tree) to connect in-game object to GUI
  - Can use `get_node()` with a **relative file path** to locate other nodes in the scene graph
  - Can use `get_tree()` to access overall SceneTree

AUDIO AND SOUND

Rez

SOUND TYPES: MUSIC

- Background score associated with a narrative work
- Dynamically change with scenes to heighten emotion
- Non-diegetic (originating outside the narrative)

Nier: Automata

SOUND TYPES: AMBIENT

- Background noises associated with narrative's scene
- Provides auditory context for scene's place and time
- Diegetic (originating inside the narrative)

SOUND TYPES: FX

- Short effects that enhance current narrative action
- Associated with particular objects and events within the scene
- Diegetic or non-diegetic

HOW TO PLAY A SOUND IN GAME?

- Sound file must be loaded and buffered before use
  - Loading and unloading sound can incur a massive performance hit if done naively
- Time of play is closely tied to player actions and dynamic events
  - Sound mixing is situationally dependent

AUDIO MIDDLEWARE

- Game engine handles:
  - Audio loading
  - Audio parameterization (e.g. attenuation, reverb, volume, etc)
  - Audio hooks into events/triggers
- Audio middleware provides GUI for sound designers to configure audio samples and events
  - Tune audio transitions, mixing, etc
  - Pass final audio and configuration parameters to engine
- Notable audio middlewares are FMOD, Wwise, and Fabric

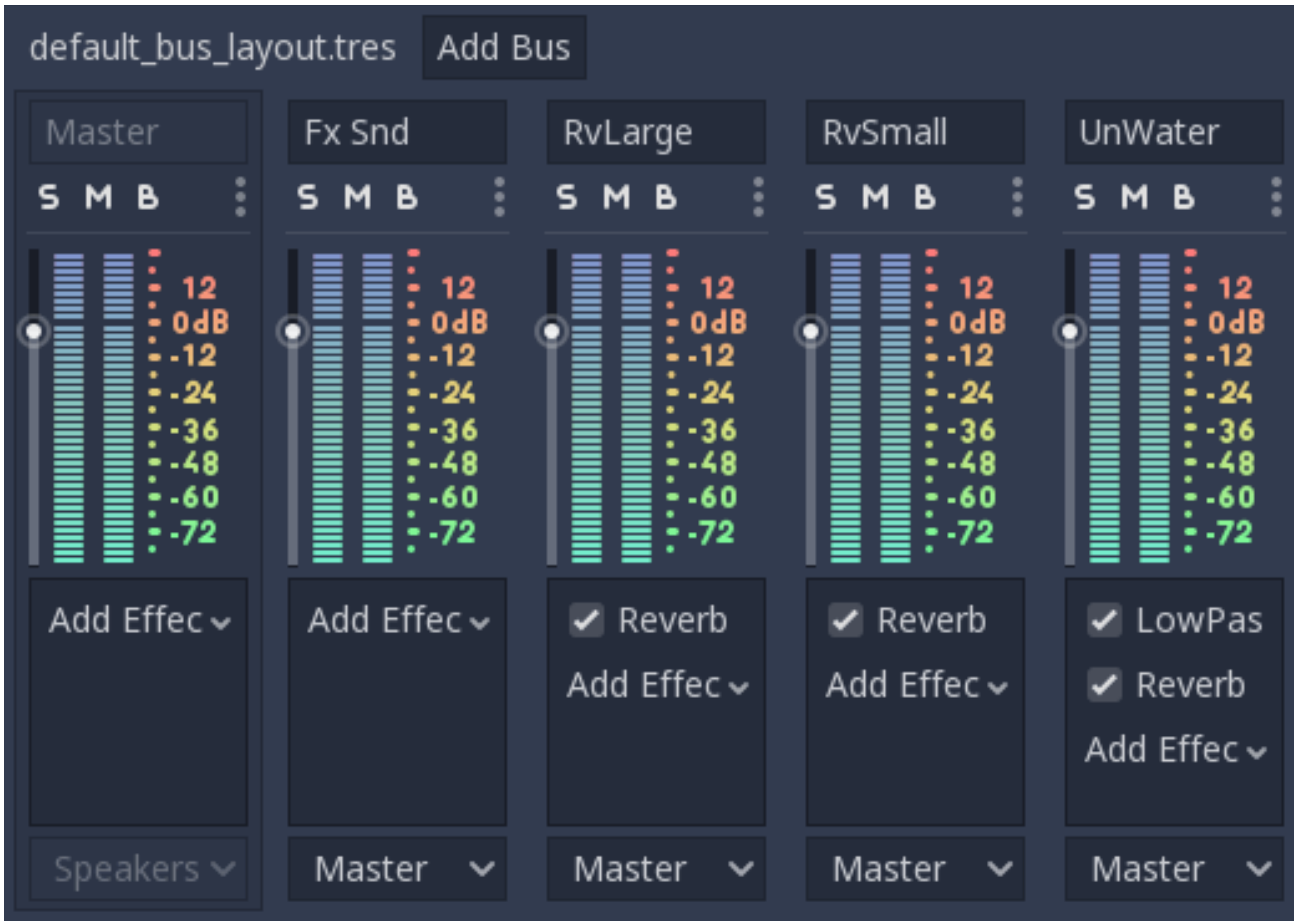
SOUND CHALLENGES

- Memory management
  - Audio file format and size tuned based on memory and performance
  - Memory leaks and/or performance issues related to audio loading/unloading
- Procedural sound
  - Avoids repetitive SFX and multiple variants of a single audio file
  - Compose SFX from collection of controllable audio components
- Quality sound
  - Audio is extremely important for player immersion
  - Usually one of the last things to be put in games

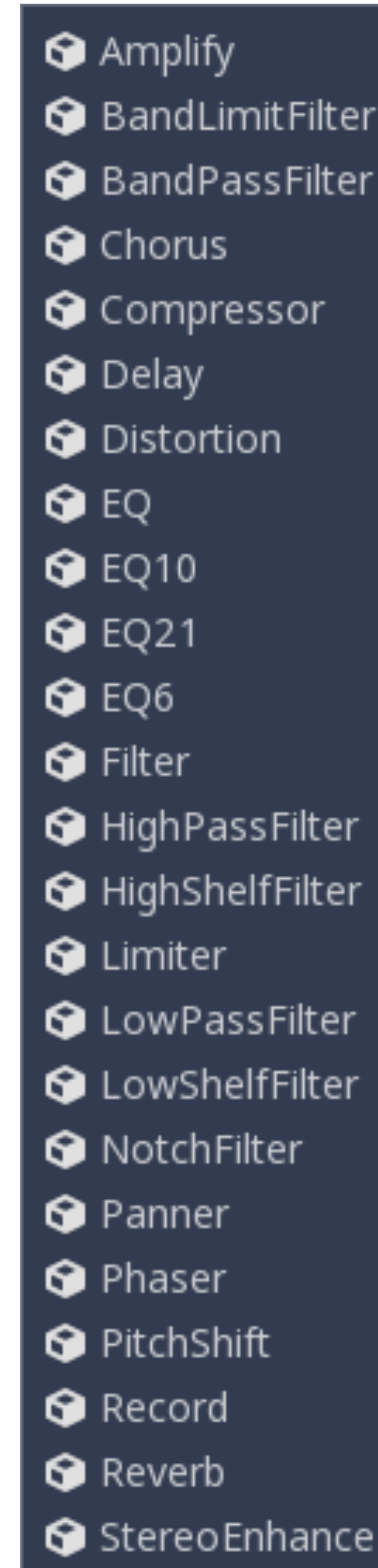
SOUND IN GODOT

- Godot processes sound via built in audio buses
- **Audio buses** channel sound to speakers
  - Audio mixed before play
  - Multiple buses allow for multiple mixes at the same time
- **Audio stream players** send sounds to buses
  - Godot currently does not support audio synthesizers
  - Must use `memnew()` constructor to create streams (or any other Object allocations in GDExtension)

GODOT AUDIO CONTROLS



Audio buses



Effects

REFERENCES

- Constraint Solvers for User Interface Layout <https://pdfs.semanticscholar.org/f95e/b57e165f7ddf779943fb05f507bef430a779.pdf>
- UI Nodes in Godot <https://docs.godotengine.org/en/3.1/getting_started/step_by_step/ui_introduction_to_the_ui_system.html>
- UI Code in Godot <https://docs.godotengine.org/en/3.1/getting_started/step_by_step/ui_code_a_life_bar.html>
- Godot Audio Buses <https://docs.godotengine.org/en/3.1/tutorials/audio/audio_buses.html>