BoneOverlay Technical Specifications

Architecture Overview

BoneOverlay is built using Unity's modern EditorToolbarDropdownToggle API, providing seamless integration with the Scene View toolbar.

Component Structure

```
BoneOverlay/
├── BoneOverlayDropdownToggle.cs    # Main toolbar UI element
├── BoneOverlayToolbar.cs           # Toolbar overlay container
├── BoneOverlayState.cs             # Persistent settings management
├── BoneDetector.cs                 # Bone detection logic
├── BoneOverlayRenderer.cs          # Visualization and interaction
└── BoneOverlaySettings.cs          # ScriptableObject (future use)
```

Key Features

Bone Detection Algorithm

1. **SkinnedMeshRenderer Bones**
    - Extracts bones array from all SkinnedMeshRenderer components
    - Includes bone weights visualization support (future)
2. **Animator Bones**
    - Supports both Humanoid and Generic rigs
    - Extracts bone transforms from Avatar definition
3. **Name Pattern Matching**
    - Patterns: "bone", "joint", "jnt", "bip", "spine", "neck", "head", "arm", "leg", "foot", "hand", "finger"
    - Case-insensitive matching
    - Hierarchical parent inclusion
4. **Duplicate Removal**
    - HashSet-based deduplication
    - Preserves hierarchy information

Rendering System

Distance-Based Filtering

- Separate distances for bones and labels
- Smooth alpha fading at distance boundaries
- Frustum culling optimization

Screen Space Calculation

```
// Perspective camera
Vector3 offsetPos = bone.position + camera.transform.right * state.SphereSize;
```

```
Vector3 edgeOnScreen = camera.WorldToScreenPoint(offsetPos);
float pixelRadius = (edgeOnScreen - screenPos).magnitude;

// Orthographic camera
float pixelsPerUnit = camera.pixelHeight / (camera.orthographicSize * 2f);
screenRadius = state.SphereSize * pixelsPerUnit;
```

Interactive Elements

- Sphere click detection using screen-space radius
- Label rendering with GUI.Button for click handling
- Hover state management

Performance Optimizations

1. **Frame-based Caching**
    - Bone detection results cached per frame
    - Distance calculations cached
2. **Culling Systems**
    - View frustum culling
    - Distance-based culling
    - LOD system for distant bones
3. **Batch Operations**
    - Minimized draw calls
    - Efficient handle rendering

API Reference

Public Properties

```
// BoneOverlayDropdownToggle
public static bool IsEnabled { get; }

// BoneOverlayState
public bool IsEnabled { get; set; }
public bool ShowLabels { get; set; }
public float MaxRenderDistance { get; set; }
public float MaxLabelRenderDistance { get; set; }
public float SphereSize { get; set; }
public float LineWidth { get; set; }
public float LabelSize { get; set; }
public Color NormalColor { get; set; }
public Color SelectedColor { get; set; }
public Color HoverColor { get; set; }
public Color LineColor { get; set; }
public Color LabelColor { get; set; }
```

Extension Points

Custom Bone Detection

```
// Future API
BoneDetector.AddCustomPattern(string pattern);
BoneDetector.RegisterCustomDetector(IBoneDetector detector);
```

Rendering Customization

```
// Future API
BoneOverlayRenderer.RegisterCustomRenderer(IBoneRenderer renderer);
```

Data Persistence

Settings are stored using EditorPrefs with the prefix `ExtEditor.BoneOverlay.`:

- Boolean values: EditorPrefs.SetBool()
- Float values: EditorPrefs.SetFloat()
- Colors: Stored as RGBA components

Unity Integration

Scene View Events

- `SceneView.duringSceneGui`: Main rendering callback
- `Selection.selectionChanged`: Updates visual state

Toolbar System

- `EditorToolbarDropdownToggle`: Main UI element
- `ToolbarOverlay`: Container for toolbar integration
- `GenericDropdownMenu`: Settings dropdown

Performance Characteristics

- **Startup Time**: < 50ms
- **Per-Frame Cost**: ~0.5-2ms (100 bones)
- **Memory Usage**: ~1MB base + 10KB per 100 bones
- **Maximum Bones**: Tested up to 1000+

Compatibility

Unity Versions

- Minimum: Unity 2022.3 (EditorToolbarDropdownToggle API)
- Tested: Unity 2022.3 - 2023.2

Render Pipelines

- Built-in Render Pipeline ✓
- Universal Render Pipeline (URP) ✓
- High Definition Render Pipeline (HDRP) ✓

Platform Support

- Windows ✓
- macOS ✓
- Linux ✓

Known Limitations

1. **Editor Only**: No runtime support
2. **Fixed Patterns**: Bone name patterns not yet customizable via UI
3. **No Filtering**: Cannot exclude specific bones
4. **Single Scene**: Works only in active Scene View

Future Enhancements

1. **Preset System**: Save/load configurations
2. **Bone Filtering**: Include/exclude specific bones
3. **Custom Patterns**: User-defined detection patterns
4. **Bone Groups**: Color-code bone chains
5. **Weight Visualization**: Show vertex weights
6. **Animation Preview**: Visualize bone movement