## Functionality Overview

The ASL System is a content creation tool that allows users to create animations for American Sign Language. Users can load signing models, animations, facial expressions, and 3D scenes, and create their own body and facial animations to compose one or more ASL sentences.

**Asset loading**

- Load and display signing models
    - Represented by an XML file that contains paths to the XNB model and morph targets.
    - Pre modeled morph targets are automatically loaded
- Load XNB animations for a character (or any character who share the same skeleton).
- Load user created animations from XML files
- Load user created expression from PNG files that have the morph target weights encoded in them.
- Load 3D scenes (XNB files).

**Playing Animations**

- Add animation clips from a list of clips to a playlist and on different layers.
    - Each animation clip contains multiple key frames for an animation
    - Each animation clip on a layer is played sequentially
    - Layers are played in parallel
    - Animation clips are displayed on the timeline.
    - Currently, layers are automatically assigned. Pre-created animations are usually assigned to the lowest BothArms layer. User created animations are currently automatically assigned to the Body layer. Also, currently only user animations are assigned to other layers than the base layer that premade animations play on. Layer blending is set to **additive** for all layers**.** Layered blending is also supported. Layered blending simply overrides any animations that were previously applied. Additive blending adds the current animation to the previously applied animation for the current frame.
- Facial animations can be created by key framing.
    - Desired morph target weights are set for a specific time and added to the list of morph target key frames. The key frames show up in the animation timeline.
- These body and facial animations can be played through by hitting play on the facial and body animation tabs.

**Creating facial expressions**

- Users can create custom facial expression by manipulating the facial controls and saving the current pose to a PNG image. User created expressions can also be loaded and applied to the signer.
    - The morph target weights are saved in the bottom row of the image. The facial controls are represented by spheres placed on the face in the regions that they affect.

**Creating custom poses and animations.**

- Users can manipulate controls placed at joints to pose a signer. The joints are rotated in the XYZ dimensions by using the X and Y axes and scroll wheel of the mouse.
- A custom animation can be created with the Pose Editor. The Pose Editor provides a timeline showing key frames for an animation. The user can pose the model and add a key frame to the timeline. Once the user is satisfied with the animation, it can be "published" to the animations playlist in the Animations Tab, and is saved to the Animations folder in the bin/content directory as an XML file.

**Creating ASL sentences with the Script Editor**

- To facilitate the process of creating an ASL sentence, users can use the script editor to alleviate the process of picking out every animation and adding it to the playlist.
- The script editor automatically compiles the ASL script, finds all the animations, and inserts them into the timeline.
- Facial animations can also be keyframed based on time in the script.

## Development functionality

**Morph target control placement**

- Morph targets can be assigned to 3d spheres placed about the head of the model. To do this, load the MorphTargetControl editor (Model -> Map Expression Controls ). This is a very crude editor. Select the morph target from the drop down, assign its movement axis (the length of the axis is how far it can move (e.g. <0, 5, 0> would move along axis <0, 1, 0> and only move 5 units away from its origin in the y direction)), and assign it a name.

**Collision skin builder**

- A tool is available to design the collision skin for the model. First enable skeleton/joint drawing. Then click the "Design CS" button. Collision skins are created from 2 joint pairs. So to create a collision mesh for the fore arm, click on the elbow joint and then the wrist joint. Once the wrist joint is selected a collision mesh will appear and it's radius can be modified by holding the right mouse button and moving in the X/Y axis. Once the collision skin is complete un-toggle the "Design CS" button.
- Collision skins can be exported and imported from the file menu.

## Language and Resources

The ASL System was developed in C# using the XNA framework. List of technologies/libraries used:

- **XNAnimation** for animation. Modified to suit the project; Added layer support and dual-quaternion skinning, and other fixes.
- **JigLibX** for collision detection (consider removing)
    - Only used at the moment to build collision skins
    - Doesn't seem to work very well with animation. It doesn't like to operate on data that it doesn't exclusively control (e.g. updating matrix data after animation and having JigLibX run collision detection on this data doesn't seem to be possible).
- **AviFile** for video capture and saving
- **MWControls** for extra windows form controls functionality. Specifically a tree view with added functionality is used in the animation playlist.
- **GREngine** for scene management and window display (My own library).
- **ANTLRWorks** for ASL script parsing and lexing, and grammar creation.
- **MS Chart Controls –** used for all timelines (a custom solution would be better though)

## Structure Overview:

ASLAuthoringTool is the main application project. It contains all the GUIs and application logic. It depends on XNAnimation, ASLAnimation (Mathsigner Animator), ASLInterpreter, and GREngine. The program operates on the state based model. Different states are responsible for different 3D windows. The MainState class is responsible for initialization of objects that are used with MainForm.cs (where user interface logic is handled) and its 3D window.

- Most important classes: **MainForm.cs, MainState.cs**

ASLAnimation is responsible for setting up the blending between different animation clips and layers with ModelAnimationController.cs. Once it has calculated the blend times for clips and layers it tells the XNAnimation library to start a cross fade between two clips for each layer. This library is also responsible for morph target blending, joint control, inverse kinematics, collision skin building, and gizmo logic.

- Most important classes**: ModelAnimationController, SkinnedMesh, JointSkeleton, CollisionSkinBuilder, Hardware/SoftwareMorphMesh, Gizmo classes**

XNAnimation is responsible for all of the acutual joint interpolation and skinning transformations, and sending the transformations to the video card.

- Most important class: **AnimationController.cs**

ASLInerpreter compiles ASL scripts and sets up animation clip playlists and sends them to ASLAnimation.ModelAnimationController.

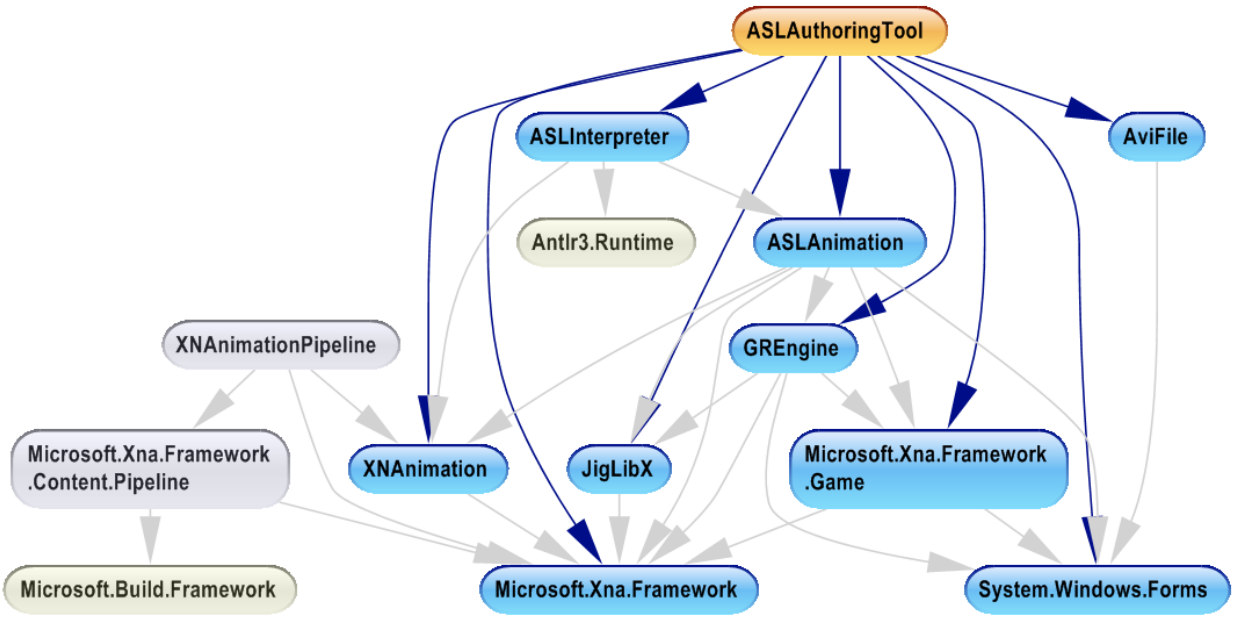
- Most important class: **SemanticAnalyzer**

GREngine is the base for most everything (except for XNAnimation). It handles graphics device setup, system initialization, updating, window control, etc.

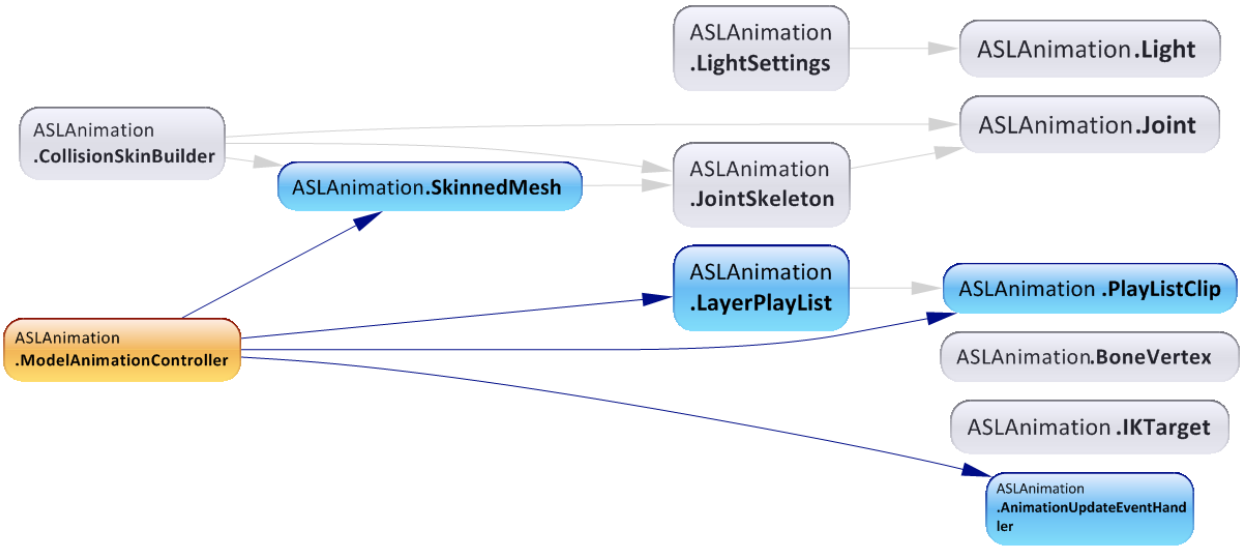
- Most Important class: **GRSystem**

## Class Dependencies

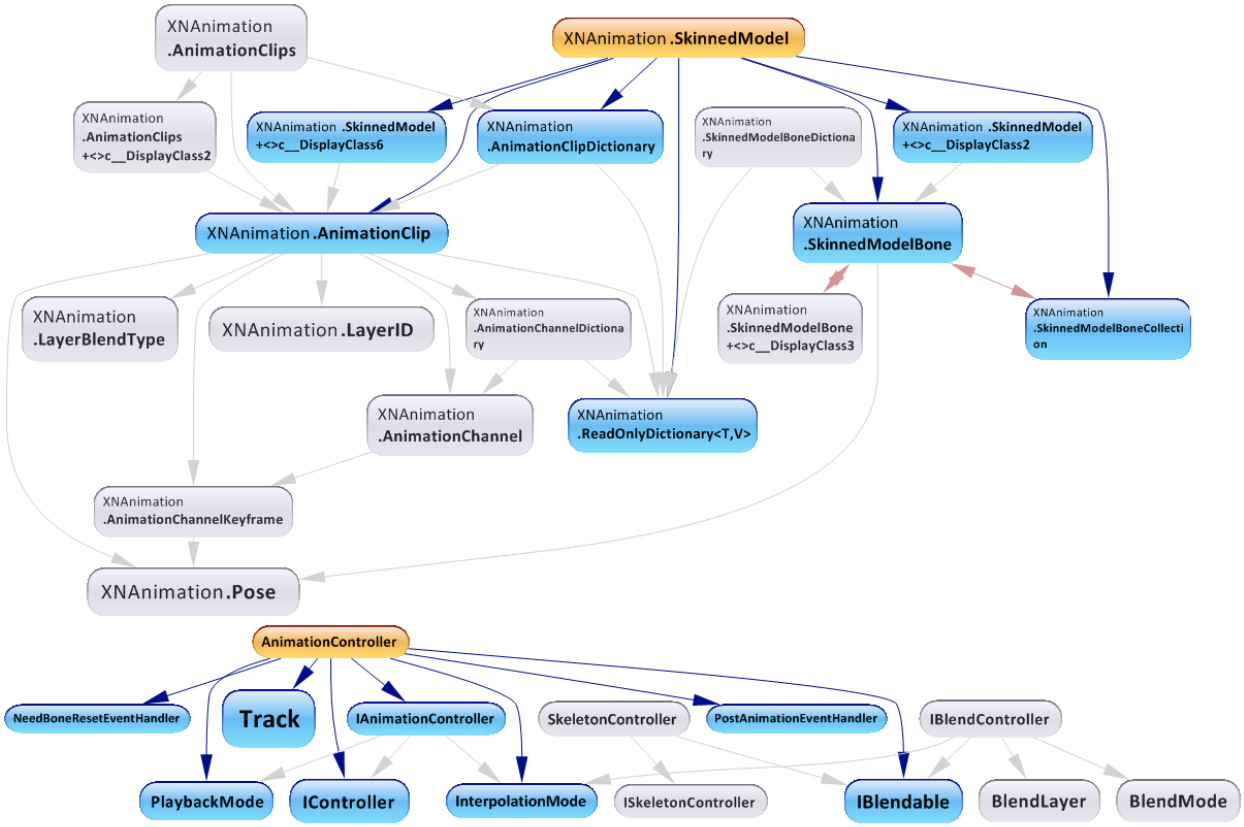
**Overall System:**



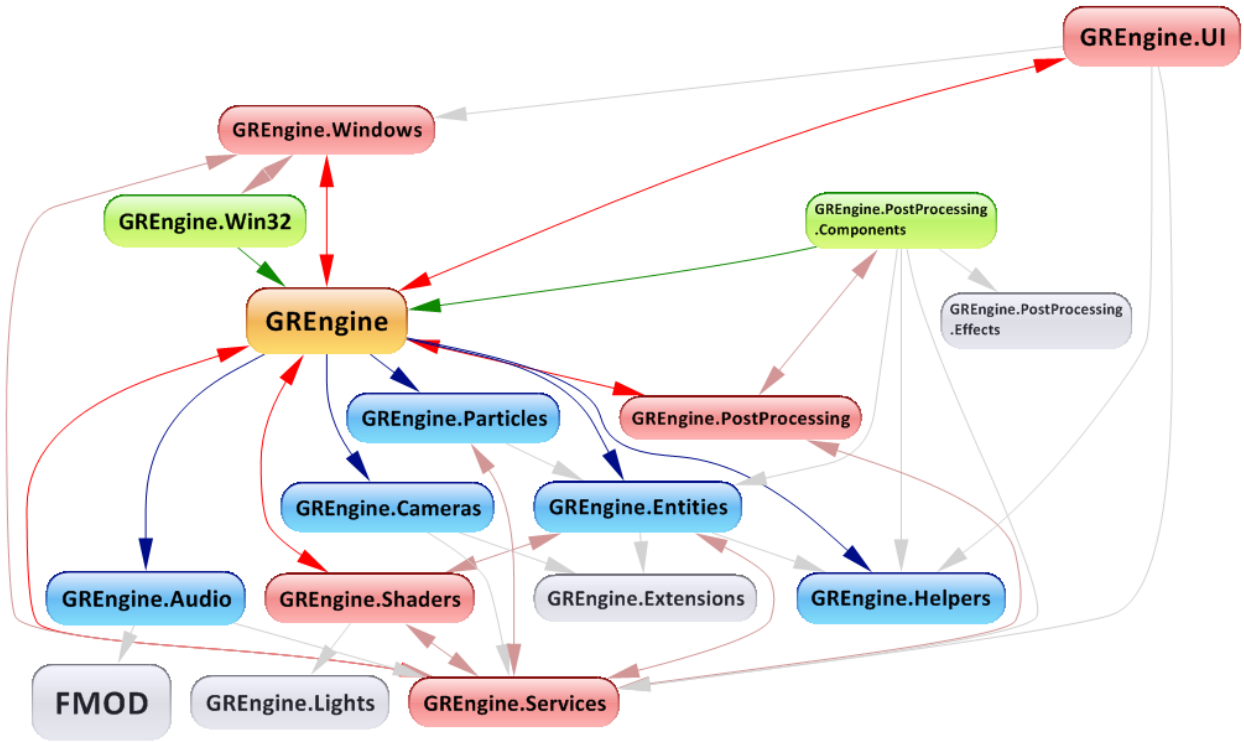

**ASL Animation:**



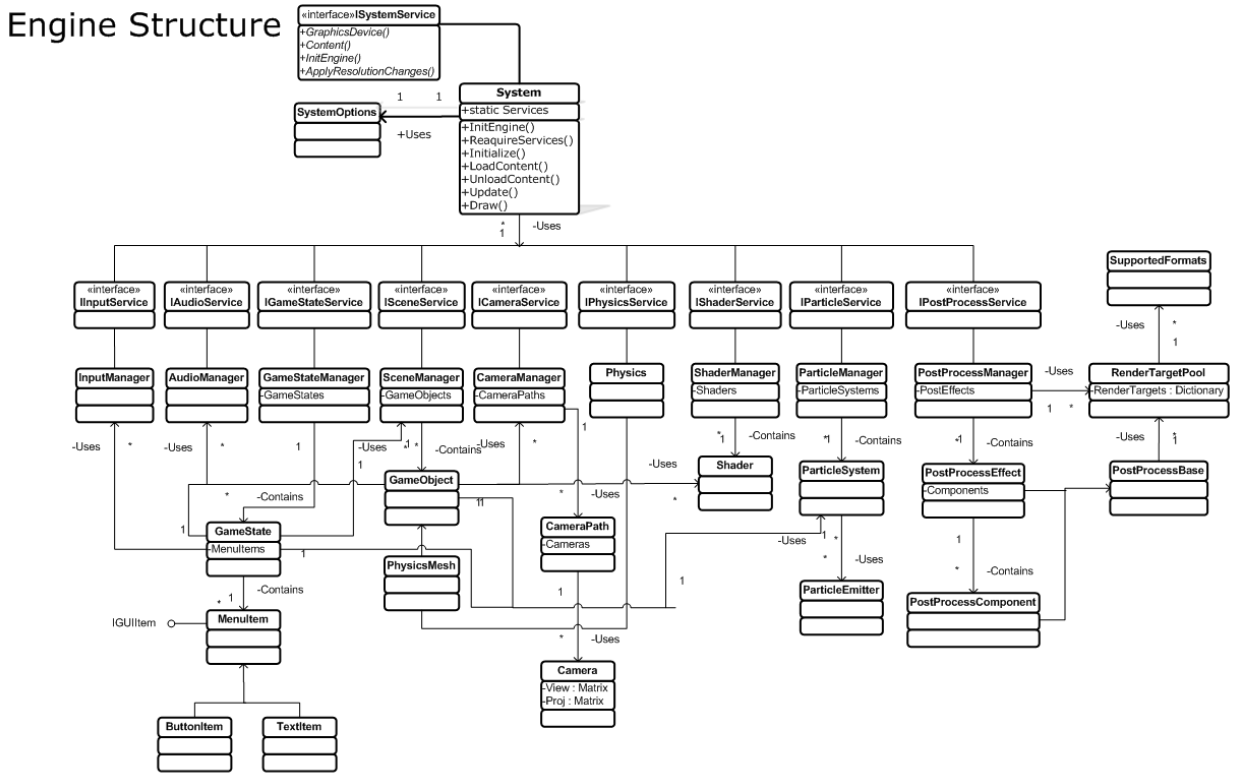
**XNAnimation:**




**GREngine:**



**Slightly dated GREngine class diagram:**




**File Formats and content pipeline**

- Artist created models and animations are exported to FBX from Maya/Max.
- FBX files are added to the content project
- **All** compiled assets are converted to XNB (includes models, textures, shaders, etc.)
- **XML –** raw XML files are loaded in at runtime and are not compiled
    - Used for model files
        - Model files point to XNB signer model
        - Contain names of morph targets (also in XNB)
    - .gzms uses XML for morph target controls/gizmos
    - .light uses XML for saving light settings (using serialization)
    - User animations use XML for saving key frames
    - Facial expressions use PNG for saving expression image and morph target weights
    - .jlpo (JigLib Physics Object) saves collision skin data in XML format (using serialization)
    - Scene files also use XML

## User Interface

- MainForm.cs
    - Main application form. Holds the 3D view, timeline, and all controls for animation.
    - Timeline shows facial animation key frames and animation clips.
        - Dragging the timeline cursor zooms in on the range contained inside the selection.
        - Crosshair on bottom left corner of timeline goes back to the previous view
    - *GUIDForm and Capture buttons used for group testing on Eurographics education paper.
- PoseEditor.cs
    - Form for creating custom user animations
    - Add key frame
    - Delete all key frames, right click a keyframe range on the timeline to delete the beginning keyframe.
    - Save Pose – inactive
    - Save Sign – saves the user animation to the bin/content/animations folder and the animations clip list box on the main form.
- ScriptEditor.cs
    - Form for entering and compiling scripts
- ExpressionEditor.cs
    - Editor for creating expressions.
    - Move controls to create an expression
    - Save expression to  a PNG