**Chosen Software Model :-**

* Skeletal Hierarchy for each animation!
* Component-Based Architecture.
* Object Orientated design.
  + Article on it : <https://www.mendix.com/blog/what-is-component-based-architecture/>
  + Based on reusable components. Each component encapsulates well defined functionality and then dropped into an application without requiring modification of other components.
    - Pros :
      * Reusability.
      * Encapsulation.
    - Cons :
      * Requires modularity which is not appropriate for every scenario.
* Makes use of frames for 2D and clips for 3D
* Ease of use was the main design philosophy along with modularity.
  + The ideal was to have it be simple enough for a developer to drop a character into the scene and then make use of the animation from the files.

**Performance Benchmarking :-**

* Here is a link to the Microsoft web page on it : <https://learn.microsoft.com/en-us/visualstudio/profiling/?view=vs-2022>
* Made use of pointers and dynamically creating instances of the animation manager and classes as it needed. So it saves on memory(Check!).
* Also often pass by reference which is technique that I picked up on from the implementation Epic games often take when making the unreal engine helping with memory efficiency.
* Frame rate in release mode stays consistent at around 60 fps.
* Measure the CPU usage and the Memory usage.
* Making use of a base class for sprite animation allowed for quite a few of the base setup functions to be defined once.
* Made large usage of maps which while slightly costly allows for easy access to specific elements. Could have used unorderd\_maps which hash keys instead to find elements.
  + However finding an element in a map is uses far fewer operations.
  + However does allow for easy mapping of data to key values.
* Made use of a lot of virtual functions.
  + Polymorphism
* Could have made an abstract base class for character.
  + Which might improve it for ease of use.
* Move some of the overlapping variables within the sprite animation to the original sprite Anim class.
* Could have limited the use of singular declaration variables.
* Could have made better use of smart pointers.

**Existing Animation Technique that could be added :-**

* Additive Blending
  + This is the creation of additive animation clips.
  + These clips contain the data that is produced when two clips are blended together for example the difference between running tired and regular running is the tired extra animation which is stored in this clip which can the be applied to other animation clips that are not the original reference animations such as walking.
  + Here’s a chapter I found that covers it in chapter 12.65 : <https://learning.oreilly.com/library/view/game-engine-architecture/9781351974271/21-9781351974271_chapter-chapter12.xhtml#sec12_6>

**Emerging Animation Technique :-**

* Soft Bodies
  + Here’s an article I found on it : <https://www.gamedeveloper.com/programming/deep-dive-the-soft-body-physics-of-jelly-car-explained>
  + It’s the squishy stretchy body of Jelly car!

A screen shot of a computer screen

Description automatically generated**CPU Usage!**

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated