# Animation & Audio

## Animations

* Sequence of images.
* In Unity we use key points. In other words if we want to move an object from point A to point B using animation we set a few key points depending on the wanted trajectory. From then on Unity creates interpolates between the points using different behaviours we choose.
* Most of the public properties of a GameObject can be animated (get their values changed in course of time).
* If you have a custom script and at some point something in it must change then we use Events. They will get looked at in detail a bit later.
* 2D animations - sequence of sprites. By changing the sprite property you can also make classic 2D animation. If you have ten sprites sequence then you grab them all and just drop them in the scene, unity will create one gameobject with animator and animation of sprite changes.
* How to make normal animation - Select the gameobject, open the Animation window and click on create button. From then on, simply move the object and double click on the timeline to create ke point.

## Avatars & Importing Animations

* Avatar - almost like the one from the movie. The idea is when you import charachter they will probably come with Avatar. Avatar is used when artist are animating a character. Instead of moving each mesh object to simulate let's a walk animation. Artist adds avatar that is rigged. Rigged means it has bone structure. This way they move the "bone structure" that is connected to the mesh. They move the arm/leg of the rigged body and the mesh glued to that bone gets changed. Unity supports the mussles that comes with the rigged body. Usually you don't need to change anything after import. (no need to use the bone structure to change anything) It is always better to handle complicated animation chnages in the source software where they got made (3DsMax, Maya, Blender etc..).
* You can check the avatar when inspecting the import settings of the character. Click on the Rig Tab -> Configure

## Animation vs Animator

* There is two aspects of making an animation in Unity:
  + Animation window presents to ability to make/edit the animation itself
  + Animator windows are for managing all animations related to an gameobject - each animation is represented by State and we go through different states using logical conditions. If X is true or B > of X then go to state Y. In the beginning there were no states and we simply had an array of animations and we used to make the logic of when to play what. Now you simply drag and drop the states and create the rules of when and what state(animation) to be active. In the animator we have parameters. These parameters are added using the UI of the Animator window then accessed by the script and managed from there. If I press the key "W" the "forward" parameter becomes True and we go to walk animation state.
  + Cool thing about the animator and states is we can adjust and control how to blend from one state to another. From idle animation to walking animation for example.

## Basic Animation import settings :

* Import Constraints - Import constraints from this asset.
* Import Animation - Import animation from this asset.
* Note: If disabled, all other options on this page are hidden and no animation is imported.
* Bake Animations - Bake animations created using IK or Simulation to forward kinematic keyframes.
  + Only available for Maya, 3ds Max and Cinema 4D files.
* Resample Curves - Resample animation curves as Quaternion values and generate a new Quaternion keyframe for every frame in the animation.
  + This option is enabled by default.
  + Disable this to keep animation curves as they were originally authored this only if you’re having issues with the interpolation between keys in your original animation
  + Only appears if the import file contains Euler curves.
* Anim. Compression - The type of compression to use when importing the animation.
* Off - Disable animation compression. This means that Unity doesn’t reduce keyframe count on import, which leads to the highest precision animations, but slower performance and bigger file and runtime memory size. It is generally not advisable to use this option - if you need higher precision animation, you should enable keyframe reduction and lower allowed Animation Compression Error values instead.
* Keyframe Reduction - Reduce redundant keyframes on import. If selected, the Animation Compression Errors options are displayed. This affects both file size (runtime memory) and how curves are evaluated.
* Keyframe Reduction and Compression - Reduce keyframes on import and compress keyframes when storing animations in files. This affects only file size - the runtime memory size is the same as Keyframe Reduction. If selected, the Animation Compression Errors options are displayed.
* Optimal - Let Unity decide how to compress, either by keyframe reduction or by using dense format.
* Animation Compression Errors - Only available when Keyframe Reduction or Optimal compression is enabled.
  + Rotation Error - How much to reduce rotation curves. The smaller the value, the higher the precision.
  + Position Error - How much to reduce position curves. The smaller the value, the higher the precision.
  + Scale Error - How much to reduce scale curves. The smaller the value, the higher the precision.

## Adding events to a Unity animation and to imported animation

* Adding events to a unity animation is easy. Click on the event icon, add one and choose a public method that belongs to a script attached to the gameobject which has animation
* Adding an event to an imported animation is happening in the import inspector where in the section Event similar to the normal way you add an event at a certain point of the animation and typing the name of the method that will be executed. Again the method should be public and part of the script that is attached to the gameobject with the animation.
* Animation Curves
* Animation curves are representing the way unity will interpolate between the key frames of the animation. There is no point in saving all positions/rotation etc between two frames when we can calculate them based on a curve algorithm. By clicking on the Curves tab in the bottom footer of the animation windows we switch to curves mode. THere we see the same time line but with the curves b/n the key frames. By right clicking on a key frame we can choose which curve to use /or to adjust the curve which represents the movement from the frame to the previous or next one.

## Audio

* Audio Sources
* Audio listeners - usually the camera, because this is where the player is usually positioned.
* Reverb Zones - makes a zone where different effects are applied. Effects like cave, concert etc..
* Audio Scripting - usual Stop(), Play()
* Audio Import main settings :
  + Load Type - The method Unity uses to load audio assets at runtime.
  + Decompress On Load - Audio files will be decompressed as soon as they are loaded. Use this option for smaller compressed sounds to avoid the performance overhead of decompressing on the fly. Be aware that decompressing Vorbis-encoded sounds on load will use about ten times more memory than keeping them compressed (for ADPCM encoding it’s about 3.5 times), so don’t use this option for large files.
  + Compressed In Memory - Keep sounds compressed in memory and decompress while playing. This option has a slight performance overhead (especially for Ogg/Vorbis compressed files) so only use it for bigger files where decompression on load would use a prohibitive amount of memory. The decompression is happening on the mixer thread and can be monitored in the “DSP CPU” section in the audio pane of the profiler window.
  + Streaming - Decode sounds on the fly. This method uses a minimal amount of memory to buffer compressed data that is incrementally read from the disk and decoded on the fly. Note that decompression happens on the separate streaming thread whose CPU usage can be monitored in the “Streaming CPU” section in the audio pane of the profiler window. Note: Streaming clips has an overload of approximately 200KB, even if none of the audio data is loaded.
  + Compression Format - The specific format that will be used for the sound at runtime. Note that the options available depend on the currently selected build target.
    - PCM - This option offers higher quality at the expense of larger file size and is best for very short sound effects.
    - ADPCM - This format is useful for sounds that contain a fair bit of noise and need to be played in large quantities, such as footsteps, impacts, weapons. The compression ratio is 3.5 times smaller than PCM, but CPU usage is much lower than the MP3/Vorbis formats which makes it the preferable choice for the aforementioned categories of sounds.
    - Vorbis/MP3 - The compression results in smaller files but with somewhat lower quality compared to PCM audio. The amount of compression is configurable via the Quality slider. This format is best for medium length sound effects and music.
    - HEVAG - This is the native format used on PS Vita. The specs of this are very similar to those of the ADPCM format.
  + Sample Rate Setting - PCM and ADPCM compression formats allow automatically optimized or manual sample rate reduction.
    - Preserve Sample Rate This setting keeps the sample rate unmodified (default).
    - Optimize Sample Rate This setting automatically optimizes the sample rate according to the highest frequency content analyzed.
  + Override Sample Rate This setting allows manual overriding of the sample rate, so effectively this may be used to discard frequency content.
  + Force To Mono - If enabled, the audio clip will be down-mixed to a single channel sound. After the down-mixing the signal is peak-normalized, because the down-mixing process typically results in signals that are more quiet than the original, hence the peak-normalized signal gives better headroom for later adjustments via the volume property of AudioSource