# Vitaboy Overview

This document gives an overall description of the VitaBoy character animation system and content pipeline.

## Introduction

VitaBoy is a skeletal 3D character animation library, that plays content authored in 3D Studio Max. Its content consists of hierarichal Skeletons, 3D texture mapped Suits that attach to the bones, and animated Skills that move the bones over time.

Artists define Skeletons in Max by using Biped, Bones Pro, or linking together normal Max objects.

Then they define Suits by attaching 3D texture mapped meshes to the bones. The meshes attached to each bone will move rigidly with the bone's coordinate system when the skeleton is animated.

Finally they define Skills by animate the skeletons with Max. The artists can use any of the tools in Max, like Biped footsteps or inverse kinematics, to animate the bones.

Once the content has been authored and saved into Max files, the artist runs the exporter to write out the Skeletons, Suits and Skills into ".CMX" files that the VitaBoy library reads. The exporter also produces ".BIN" files containing binary floating point translations and rotations, and ".MSH" files containing 3D meshes. The ".MSH" files reference the texture maps by name, but the exporter does not write them out, so the correct bitmap files must be put into place for the VitaBoy to find.

## Architecture

The VitaBoy library is divided into two main source code files, which contain the device independant part (skeleton.cpp), and the device dependant part (vitaboy.cpp). This is designed to make it easy to port VitaBoy to different 3D libraries and embed it in different applications, by changing only the device dependant vitaboy code.

The device independant skeleton code doesn't know anything about the device dependant vitaboy code, but the skeleton is hooked together with it through a set of callbacks and void data pointers.

## Strucure of VitaBoy Content

- Skeletons
  - Bones
    - 3D Translation: X, Y, Z
    - 3D Rotation: X, Y, Z, W
- Suits
  - Skins
    - Meshes
      - Texture Maps
- Skills
  - Motions
    - Translation Streams
      - 3D Translation: X, Y, Z
    - Rotation Streams
      - 4D Rotation: X, Y, Z, W
    - Event Streams
      - Properties
        - Key Names
        - Value Strings

## Definitions

- ### Animation Database

  A Microsoft Access database with fields that describe the specifics of each animation. 3D Studio Max uses MaxScript to read the Animation Database via OLE Automation. The database makes it possible for the exporter to know about all the content, where the Max files are, and where they should be exported, so as to eliminate human error and to make batch exporting possible.

- ### 3D Studio Max R2

  Version 2 of 3D Studio Max, an extensible 3D authoring tool.

- ### Biped

Part of the Character Studio plug-in for Max, used to create and animate two legged skeletons, by using high level controls like footsteps and inverse kinematics.

- **MaxScript**

  An scripting language that plugs into 3D Studio Max R2. MaxScript can be used to create utility panel user interfaces like the CMX Exporter, to automate tasks like batch exporting, and to extend Max to do all kinds of cool stuff.

- **CMX Exporter Utility Panel**

  A user interface to the CMX exporter, written in MaxScript, driven by the Animation Database, and driving SourceSafe.

- **CMX Files**

  VitaBoy text file format, containing Skeletons, Suits, and Skills. The Suits refer to external meshes in binary MSH Files, and the Skills refer to external floating point numbers (representing translations and rotation) in binary BIN files.

- **MSH Files**

  DDD binary file format, containing 3D geometry and texture names. The names can be standard color names (defined by a table compiled into DDD), or the names of bitmap files (sans ".BMP" suffix) from the game's "Textures" directory.

  The mesh textures get their names from the 3dsmax material names. In 3dsmax it takes two nested materials to map a bitmap texture, and both of them should have the same name: the name of the bitmap sans ".BMP". The bitmap is not exported, just its name, so it's the responsibility of the artist to check the textures into the game's "Textures" directory.

- **BIN Files**

  VitaBoy binary file format, containing raw floating point numbers. They represent the streams of 3D translations and 4D rotations, used by the skills to animate the skeletons.

# Procedures

- ## Authoring Skeletons
  - Naming conventions.
  - Biped skeletons.
  - Extending Biped skeletons.
  - Constructing other skeletons from scratch.
  - Tagging skeletons.
  - Set standard position, direction, and pose.
  - Save Max file as prototype.

- ## Authoring Suits
  - Merge in template skeleton or start with template max suit file.
  - Create low-poly meshes, with bitmap texture map materials.
  - Name materials the same as texture map bitmaps.
    - Textures are not exported, just the names of the materials, which should be the same as the texture map bitmap file names, sans ".BMP".
  - Attach meshes to skeleton.
    - It's possible to attach several meshes to one bone, and attach meshes to each other. Each mesh attached to the Skeleton is exported relative to the bone to which it or its ancestor is attached.
  - Tag up Suits on Skeleton with "suit=name".
  - Multiple suits in the same Max file.
    - Different suits on different skeletons.
    - Different suits on the same skeleton at different times.
    - Different suits on different bones of the same skeleton.
  - Export suit.
    - You have to check the exported content out of SourceSafe before you re-export it, or else it will be read-only and the exporter will complain. The exporter utility has a "Check Out" button that will automatically do that for you.
  - Check results into SourceSafe, including CMX, MSH, BMP and MAX files.
    - Exporter has a "Check In" button to check in CMX and MSH files.
    - Artists must manually check BMP textures into the appropriate SourceSafe directory, and check MAX files back into SourceSafe

content directory.

- **Authoring Skills**
    - Merge in template skeleton or start with template max skill file.
    - Tag up Sklls on Skeleton with "beginskill=name" and "endskill=name".
    - Arguments to beginskill.
        - moving
        - xoffset
        - yoffset
        - rotation
        - includebones
        - excludebones
        - ...
    - Tag events.
        - interruptable
        - footstep
            - Can be on root, not necessarily on toe.
        - anchor
            - Events on toes to be anchored.
        - xevt
        - dress
        - undress
        - censored
        - sound
        - selectedsound
        - deselectedsound
        - ...
    - Multiple animations in the same Max file.
        - Different skills on different skeletons.
            - Used for multi-person interactions like kissing.
            - Use xoffset, yoffset, rotation to relativize skeletons.
        - Different skills on the same skeleton at different times.
            - Used to make one skeleton perform several
            - skills one after the other.
        - Different skills on different parts of the same
        - skeleton at the same time.
            - Used to export both full and partial body versions of one animation, or to split an animation up into left and right side partial body skills, etc.
        - Different skills on the same parts of the same

        - skeleton at the same time.
            - Used to export one animation as several skills with different properties or events.
    - Special Types of Animation
        - Events
            - Tags not recognized by the exporter are passed through to the skill as events on bones in time.
            - Events are usually on the root bone, but they can occur on any bone, like anchor events on the toe bones.
            - The event format is "name=val". You can have several events on the same bone at the same time, by typing each on a different line in the same note.
            - Synchronize animations with tree code using "xevt=#", which passes the numeric argument down to the tree code playing the animation.
            - You can tag a note track with "skill=name", and the events in that track will only be seen by the skill of that name. This is so you can tag up several skills from the same animation, but with different events.
        - Looping
            - The frame after the last frame should be the same as the first frame.
        - Walking
            - Moving
                - "moving" tag
                - Conventions
                    - Starting foot
                    - Loop cut point
                - Anchors
                - Interruptability
                - Footsteps
            - Turning
                - "absolute" tag
            - Adjusting
                - "absolute" tag
            - Start Walking
                - "absolute" tag
        - Reaching
            - Standard begin and end poses.
            - Events for pick up/drop (xevt=?).
            - Reversable?
        - Offset origin

            - Get in/out of chair, etc.
            - Use xoffset, yoffset, rotation to establish the skeleton's local coordinate system.
        - Partial body
            - Put beginskill/endskill tags on a bone beneath the root. Use includebone/excludebone to add or subtract bones from skill.
        - Mixing Animations
            - Animations can be mixed together during playback. The transition from one animation to the next can be smoothed by fading out the previous skill (instead of stopping it abruptly) when a new skill starts playing. This makes it easier for the artist, so it's not necessary to make the begin and end poses of every animation match exactly.
            - It's possible to blend together two or more synchronized skills by cross fading. So certain animations must be designed and authored to be mixed in special ways.
            - For example, you could take an animation of the head looking up and turning left to right, and another of the head looking down and turning left to right. Then you could combine them to look in any direction, by changing the time to look left and right, and cross fading from one to the other to look up and down.
            - Or you could blend together a long walk cycle and a short walk cycle in different proportions, to get footsteps of any distance beween the two extremes. This requires that the walking animations be synchronized, the same duration, with footsteps and events at the same time.
    - Export skill.
        - You have to check the exported content out of SourceSafe before you re-export it, or else it will be read-only and the exporter will complain. The exporter utility has a "Check Out" button that will automatically do that for you.
    - Check results into SourceSafe, including CMX, BIN and MAX files.
        - Exporter has a button to check CMX and BIN files in.
        - Artists must manually check MAX files back into SourceSafe content directory.

- ## Export Process

    - Installing 3D Studio Max 2
        - Get the latest version of 3D Studio Max. If you want to compile the CMX Exporter, you also need to copy the MaxScript SDK

"mssdk" from the 3dsmax2 distribution CD to "3dsmax\mssdk".

- Upgrading MaxScript Extension Language
    - MaxScript is a standard scripting language plug-in included with 3DSMax2. The latest version has fixes to OLE automation database access bugs. Get the latest version of the MaxScript plug-in from the CMX exporter distribution, and put it into "3dsmax\sdtplugs\???.???".
- Installing CMX Exporter MaxScript Extension Plug-In
    - Get the latest version of the CMX exporter plug-in from the CMX exporter distribution, and put it into "3dsmax\plugins\???.???". It extends the MaxScript language by adding primitives for note track access, environment access, and the CMX exporter support.
- Using the CMX Exporter Utility Panel
    - Get the latest version of the MaxScript code, and put it in "3dsmax\scripts\startup\???.ms", so it's automatically loaded when you start up MaxScript from the Max utility panel.
    - The MaxScript code reads in the animation database from Access (via OLE automation), and implements a utility panel user interface to the CMX exporter.
    - You can open up the CMX Exporter utility panel from the Utilities menu in the MaxScript plug-in panel.
    - The utility panel has scrolling list of animations, so you can click to select any one, load the max file, or export the Cmx file.
    - It has "Check In" and "Check Out" buttons that automatically invoke Source Safe, to check exported files in and out of source control.
    - It has buttons to batch export every animation in database, or every Max file in list of file names.
    - There is a write-enable check box that can be un-checked to disable the exporter writing to disk, for testing and validity checking purposes.
    - You can press the "Open Scripter" (???) button on the MaxScript utility panel, and a text window will open up. In it that window, the exporter prints all kinds of informative messages showing what it's doing, and error messages about anything that goes wrong.

- ## Testing Animations

    - TDS Object Animation Previewer
        - Menu of object animations to play on selected character.
    - TDS Global Animation Previewer

        - Menu of global animations to play on selected character.
    - Edith Animation Inspector
        - Control panel with scrolling lists of all loaded skeletons, suits, and skills.
        - Put any suit on any skeleton.
        - Play any skill, adjusting all parameters.
    - Debugging SAnimator Walking
        - Speed controls
        - Show routing goals
        - Show anchor position