Extreme Motion SDK - Overview

Extreme Reality

9/10/2014

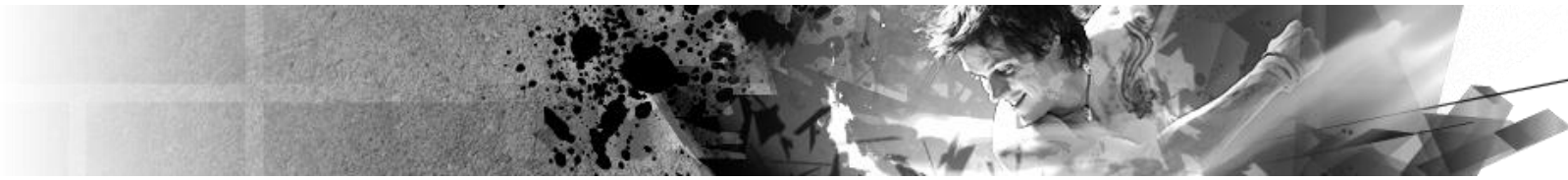


Table of Contents

1 About This Document

The following document provides a brief overview of the Extreme Motion SDK. The document describes the main outputs of the SDK and its state machine.
This document is intended for any developer who wants to start working with Extreme Motion and wants to understand the basics of the SDK.
Extreme Reality welcomes your suggestions for improving our documentation. If you have comments, send your feedback to support@xtr3d.com.

Other SDK documents and resources include:

- "ExtremeMotionSDK_EULA.html"
- "ExtremeMotionSDK_ReleaseNotes.html"
- "Extreme MotionSDK_API_Reference.html"
- "Powered By Extreme Reality" splash screens

2 What is Extreme Motion?

Extreme Motion SDK is a SW only middleware which uses any standard 2D camera to extract the skeleton of the user standing in front of the camera. The middleware analyzes the streamed video in real-time and outputs the 3D coordinates of the skeleton joints. Optimal conditions are when the user is between 4 to 13 feet (1.5m – 4m) from the camera.

3 Supported Cameras and Resolutions

Extreme Motion SDK supports all standard cameras. At any given time, only one webcam will be used by the Extreme Motion SDK. If more than one camera is available, the SDK automatically selects the camera it will work with, following this logic:

- PCs & Laptops - If a peripheral camera is connected to the device, it will be selected by default. If no external camera is connected the SDK will selects the native camera.
- For smartphones and tablets the SDK always selects the front facing camera.

Extreme Motion SDK captures 640 x 480 RGB images from a camera, processes them and extracts the skeleton coordinates of the user. The camera may support higher resolutions.

Extreme Motion SDK does not support portrait mode as the camera's horizontal field-of-view is considerably reduced in this mode.

Developers may use their own image acquisition module and inject the captured 640 x 480 RGB images to Extreme Motion SDK.

4 Supported Devices and Operation Systems

The SDK supports Win32, WinRT, iOS and Android.
Handheld devices (smartphone & tablets) are only supported in landscape operation mode.
For full specification, please refer to the "Compatibility Specifications" document at:
http://www.xtr3d.com/developers/resources/

5 Processing Frame Rate

The processing frame rate of the SDK is bounded by the camera's frame rate and is influenced by the following:

- CPU – the stronger the CPU, the higher the frame rate will be
- Lighting conditions – in low lighting conditions the camera's frame rate is lower

6 Joints

6.1 Detected Joints

The engine outputs the coordinates of the following joints in each frame:

- Head
- Shoulders Center
- Spine
- Hip Center
- Shoulder Left
- Shoulder Right
- Elbow Left
- Elbow Right
- Hand Left
- Hand Right

6.2 Joints States

Every joint has a tracking state per camera frame. The tracking states are as follows:

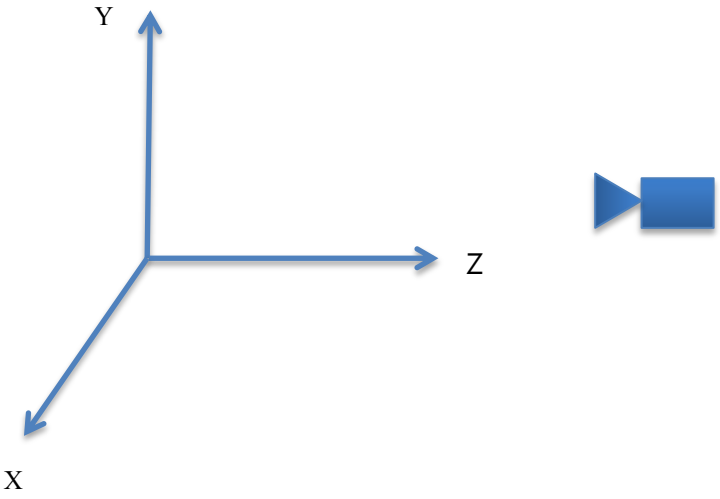
- **Not Tracked**: Joint was not detected
- **Inferred**: Joint is approximated because it is occluded, out of camera's field of view, or due to noisy conditions
- **Tracked**: Joint is detected

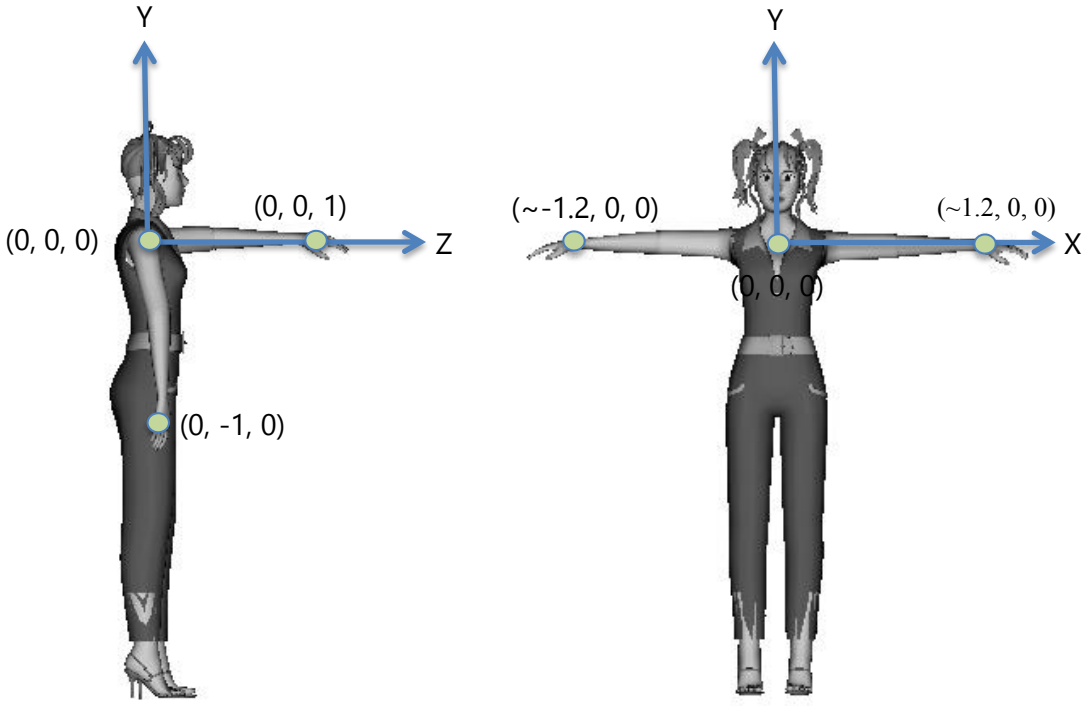
6.3 Joints Relative Coordinate System

The three dimensional Cartesian coordinate system of the joints is composed of the 3 axes X, Y, and Z. In each frame, the origin of the coordinates system (0, 0, 0) is set to the middle of the line

connecting the shoulders. The coordinate system measurement unit is “user arm length.” A vector of length 1 is equal to the length of the user’s arm.

The following diagrams illustrate the coordinate system:





6.4 Joints Image Coordinates

Each joint has, in addition to the X, Y, and Z coordinates, two additional coordinates which represent the location of the joint in the image – imgCoordNormHorizontal and imgCoordNormVertical.
These coordinates are in the range [0-1] and are relative to the width and the height of the image. For example, an image of resolution 640x480 where the imgCoordNormHorizontal = 1, results in an X location of the joint in the image of 639.

The following is an example showing how the image coordinates of the left hand are calculated:

Image width = 640
Image height = 480
Hand coordinates in the image (x,y) = (470, 120)
imgCoordNormHorizontal = 470/640 = **0.73**
imgCoordNormVertical = 120/480 = **0.25**

7 Gestures

Gestures are certain movements or static poses of the skeleton, which are used to trigger application events like clicking a button, or striking an enemy in the game.
There are two ways to detect gestures in game:

- Xtr3d sdk supports some gestures out-of-the-box. The developer can use xml file to choose a set of gestures relevant to his application.These gestures include, for example, hand-swipe to the left/right, head-movement and static positions like “hands in T-Pose”. *SuperPose* is a GUI tool which allows easy configuration and test the gesture in real-time, without a need to write the xml file manually. You can download it under http://www.xtr3d.com/developers/resources/

- The developer can detect any gesture using custom application-code, using the joints positions. For example: to detect a gesture of T-Pose where the user holds both his hand spread out horizontally in shoulder height (hence “T” like pose), the developer will calculate the right and left hand angles and check that they are close to 90 degrees.

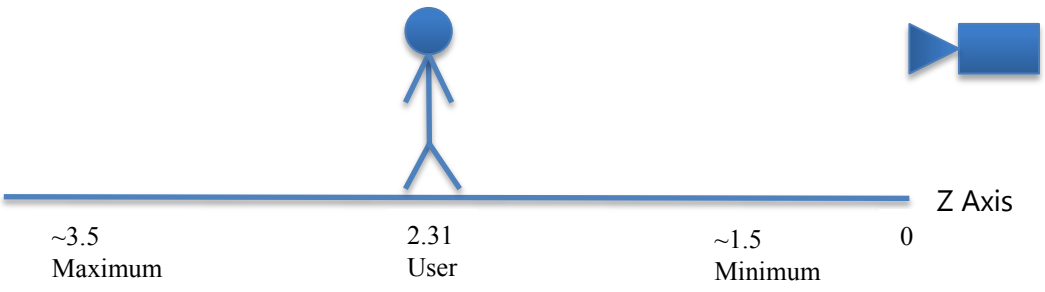
8 Proximity Measurement

The proximity measurement represents the approximate distance of the user from the camera and is a linear function of the distance in meters of the user from the camera. The proximity is calculated according to the user's body dimension. When the user's body dimension is becoming larger the user is moving closer to the camera, and when the user's body dimension is becoming smaller the user is moving away from the camera.

9 Activity Range

The engine can track the user accurately in a specific range along the Z axis. This range is called the *activity range*, and is defined by the approximation distance. If the user is out of the activity range, a warning is triggered indicating if the user is too close or too far from the camera.

The following figure illustrates the activity range:



10 Licensing

There are 2 types of license files.

- **Temporary License – EM_Trial.lic**
  This is the standard file provided with the SDK.
  It is a fully functional license, which can be used with any application.
  However, the license is time limited – For 60 days since its creation (Hence 60 days since the SDK is published on the website).

  If the more than 60 days are required for development and testing phase, it is always possible to download a new version of the SDK and enjoy extra 60 days.

- **Productization license**
  This license is a perpetual license and provided by Extreme Reality upon specific requests, for developers wishing to make their product available to the public.
  Please fill in the Productization Form, in order to obtain such a license.

If you fail to initialize Extreme Motion SDK with an Invalid License error, make sure that the license is located where Extreme Motion binary files are located. If you are using a trial license, make sure that it has not expired.

11 Skeleton States

11.1 Init State

Init state is the first state of the skeleton. The duration of this state depends on the device CPU.

11.2 Calibration State

In the *Calibration* state, the engine is searching for a person performing the activation gesture, which is how the user signals the engine that they want to take control and that they are the one to be tracked. The activation gesture is illustrated in the following image:

11.3 Tracking State

The engine enters the *Tracking* state after the user performed the activation gesture successfully. During the tracking state the engine output is comprised of the coordinates of the user's joints. If the user face is not seen by camera for more than 4 seconds the engine returns to the activation state.

11.4 Not Tracking State

The *Not Tracking* state indicates that no user is being tracked.
After 3 consecutive seconds in *Not Tracking* state, the engine restarts and goes back to *Init* state.

12 Warnings

The engine can output the following warnings:

- Too many people in the background
- Backlight lighting is too strong
- Low light
- User is too close to the camera
- User is too far from the camera
- User is too close to right side of the camera's FOV
- User is too close to left side of the camera's FOV

12.1 Too Many People in the Background

The engine outputs the "too many people in the background" warning during the activation state when there is more than one person in the background.

12.2 Backlight is Too Strong

The engine outputs the "Backlight is too strong" warning when there is a strong light source behind the user.

12.3 Low Light

The engine outputs the "Low light" warning when there in not enough light.

12.4 User Is Too Close To the Camera

The engine outputs the "too close" warning when the user is too close to the camera and his distance approximation is lower than the minimum allowed. In such cases, the user should move backward.

12.5 User Is Too Far From the Camera

The engine outputs the "too far" warning when the user is too far from the camera and his distance approximation is higher than the maximum allowed. In such cases, the user should move forward.

12.6 User is Too Close to the Right Side of the Camera's FOV

The engine outputs the "too close to the right side of the camera's FOV" warning when the user is too close to the right side of the camera's FOV. In such cases, the user should move to the center.

12.7 User is Too Close to the Left Side of the Camera's FOV

The engine outputs the "too close to the left side of the camera's FOV" warning when the user is too close to the left side of the camera's FOV. In such cases, the user should move to the center.

13 Best Practices

13.1 Environment

Extreme Motion works in a wide variety of environments, both indoor and outdoor.
Below are some tips for optimal skeletal detection:

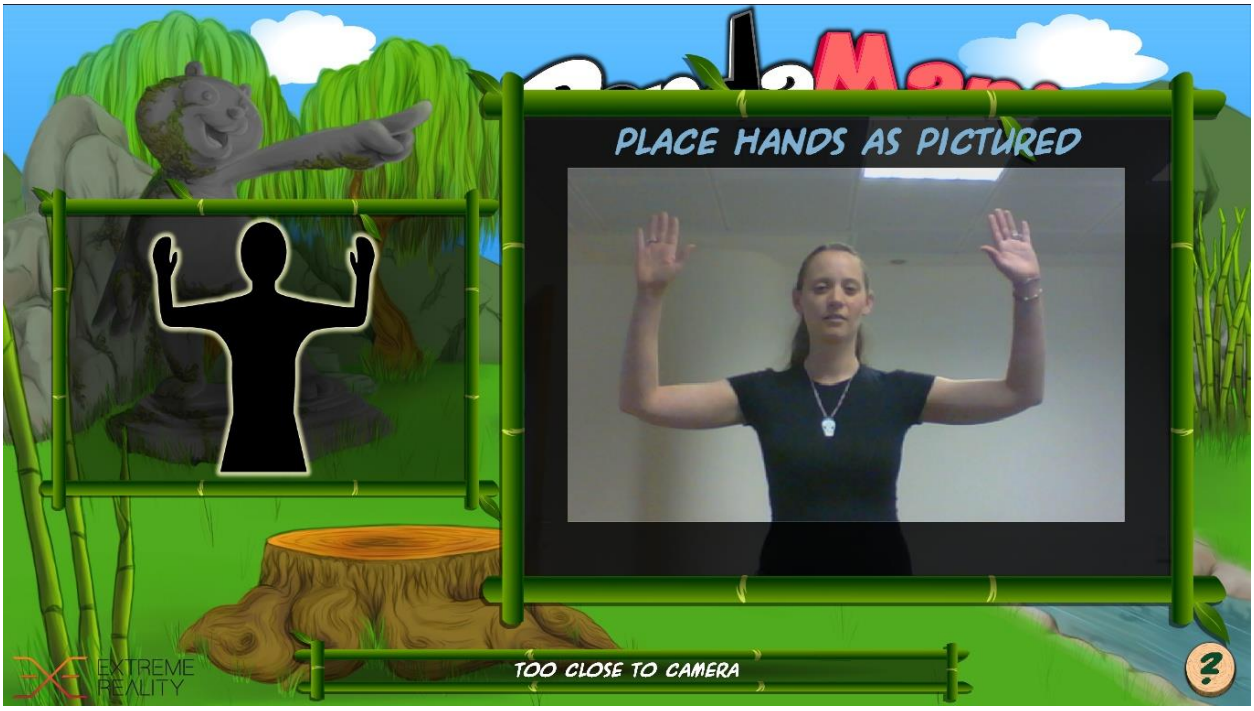
- Avoid direct sunlight / Light in the camera
- Avoid completely dark environments
- Avoid having many people surrounding the person being tracked
- Avoid changing, messy and colorful backgrounds. Prefer clean single-colored background

- Clothing: for the most accurate results, short-sleeves shirt, pulled back hair and a single-colored shirt will give the optimal results. Extreme Motion SDK works well for games, even without these conditions, but if you have a controlled environment, these are the recommended settings

13.2 Activation

The activation stage, which corresponds with the Init and Calibration states (whether entered for the first time or after the engine has reset), should be used to give feedback to the user about how the camera should be positioned, and therefore, it is recommended in the activation screen to show the RGB video streamed from the camera.

The following picture shows the activation screen in the PandaMania game:



13.3 Pause

For applications in which it is insufficient to pause only when the user is not being tracked any more, the application should have a unique and intuitive gesture for pause.

The following are two examples for positions which can be used:

13.4 Navigation in Menus

The sample application UIConcepts demonstrates several concepts for navigation in menus and items selection. From the application's main window one can select a specific UI concept. After selecting one, a new window is opened, which is controlled according to the selected concept. You can download the tool under: http://www.xtr3d.com/developers/resources/

14 Sample Applications

Sample	Description	Programing Language	OS
VisualSkeletonSample	Application showing the skeleton data on top of the acquired video	Unity	Windows, iOS, WinRT, Android
UIConceptsSample	Demonstrates several UI navigation concepts	Unity	Windows, iOS, Android
ProximitySample	Demonstrates the proximity concept	Unity	Windows, iOS
CSharpVisualSkeletonSample	Application showing the skeleton data on top of the acquired video	C#	Windows
CConsoleSample	Console application using the C API	C	Windows
ObjecticeCBasicSample	iOS native application using the C API	Objective C	iOS
CSharpBasicSample	Shows how to call the SDK from WinRT	C#	WinRT
SkeletonJointsSample	Shows how to call the SDK using Java	Java	Android