felixjones / pmx21.md
Last active 2 hours ago

PMX (Polygon Model eXtended) 2.0, 2.1 File Format Specifications

pmx21.md

PMX (Polygon Model eXtended) 2.1

This is an English description of the .PMX file format used in Miku Miku Dance (MMD).

PMX is the successor to the .PMD format (Polygon Model Data).

This is work-in-progress! Please leave feedback in the comments.

Todo

If you are able to help with any of these, please do! Either fork and request a merge or write in the comments below what needs to be done.

- Figure out how SDEF uses C, R0, R1
- A lot of unknown/incomplete information in the Bone section onwards

Read Format

- Must be read in binary mode
- Format is Little-Endian throughout
- Text encoding is UTF16LE or UTF8 depending on the global encoding setting

Types

C Types

Name	Size (bytes)	Range	Notes	std type
byte	1	-128..127	char	int8_t
ubyte	1	0..255	unsigned char	uint8_t
short	2	-2^15..(2^15 - 1)	short	int16_t
ushort	2	0..(2^16 - 1)	unsigned short	uint16_t
int	4	-2^31..(2^31 - 1)	int	int32_t
uint	4	0..(2^32 - 1)	unsigned int	uint32_t
float	4	IEEE 754 single	float	float

PMX Types

Name	Size (bytes)	Structure	Notes
vec2	8	float, float	XY vector
vec3	12	float, float, float	XYZ vector
vec4	16	float, float, float, float	XYZW vector
text	4 + length	int, byte[]	Byte sequence encoding defined in Globals
flag	1	byte	8 flags per byte. 0 = false, 1 = true
index	1/2/4	byte/ubyte/short/ushort/int	Type defined in file header, sign depends on usage

Index Types

Name	Type 1	Type 2	Type 4	Type 1 limit	Type 2 limit	Type 4 limit	Nil Value
Vertex	ubyte	ushort	int	255	65535	2147483647	N/A
Bone	byte	short	int	127	32767	2147483647	-1
Texture	byte	short	int	127	32767	2147483647	-1
Material	byte	short	int	127	32767	2147483647	-1
Morph	byte	short	int	127	32767	2147483647	-1
Rigidbody	byte	short	int	127	32767	2147483647	-1

File Structure

Header		Ver
Model information		2.0
[int] Vertex count	Vertices	2.0
[int] Surface count	Surfaces	2.0
[int] Texture count	Textures	2.0
[int] Material count	Materials	2.0
[int] Bone count	Bones	2.0
[int] Morph count	Morphs	2.0
[int] Displayframe count	Displayframes	2.0
[int] Rigidbody count	Rigidbodies	2.0
[int] Joint count	Joints	2.0
[int] SoftBody count	SoftBodies	2.1

Header

Field	Structure	Value	Notes
Signature	byte[4]	"PMX " [0x50, 0x4D, 0x58, 0x20]	Notice the space at the end
Version	float	2.0, 2.1	Compare as floating point values

Field	Structure	Value	Notes
Globals count	byte	Fixed at 8 for PMX 2.0	This could be variable
Globals	byte[N]	N = Globals count	See Globals below
Model name local	text	Name of model	Usually in Japanese
Model name universal	text	Name of model	Usually in English
Comments local	text	Additional information (license)	Usually in Japanese
Comments universal	text	Additional information (license)	Usually in English

Globals

Index	Name	Values	Effect
0	Text encoding	0, 1	Byte encoding for the "text" type, 0 = UTF16LE, 1 = UTF8
1	Additional vec4 count	0..4	Additional vec4 values are added to each vertex
2	Vertex index size	1, 2, 4	The index type for vertices (See Index Types above)
3	Texture index size	1, 2, 4	The index type for textures (See Index Types above)
4	Material index size	1, 2, 4	The index type for materials (See Index Types above)
5	Bone index size	1, 2, 4	The index type for bones (See Index Types above)
6	Morph index size	1, 2, 4	The index type for morphs (See Index Types above)
7	Rigidbody index size	1, 2, 4	The index type for rigid bodies (See Index Types above)

Vertex

Additional Vec4s

Each vertex may gain up to 4 additional vec4 values as defined in the Globals list in the header. Usage tends to vary as most applications will simply pass the values to a vertex shader and let the shader decide the usage.

As an example, additional vec4 *may* be used for the following;

- Additional texture UV
- Specular map UV
- Normal map UV
- Vertex colour (PMX 2.1 has a material extension that uses the first additional vec4 for this)

As they are implementation dependant it is most-likely safe to ignore additional vec4, however this isn't a guarantee, a PMX model can be produced expecting additional vec4 to be used in a specific way.

Deform Types

Each vertex can be tied and weighted to bones based on the Bone Index (See Index Types above). A single vertex can be tied to a maximum of 4 bones.

There are multiple types of deforms, each one varies in length and usage.

The bone index of -1 is a nil value, the bone should be ignored.

BDEF 1

Version 2.0

type	name	Notes
index	Bone index	Weight = 1.0

BDEF2

Version 2.0

type	name	Notes
index	Bone index 1	
index	Bone index 2	
float	Bone 1 weight	Bone 2 weight = 1.0 - Bone 1 weight

BDEF4

Version 2.0

type	name	Notes
index	Bone index 1	
index	Bone index 2	
index	Bone index 3	
index	Bone index 4	
float	Bone 1 weight	Weight total not guaranteed to be 1.0
float	Bone 2 weight	Weight total not guaranteed to be 1.0
float	Bone 3 weight	Weight total not guaranteed to be 1.0
float	Bone 4 weight	Weight total not guaranteed to be 1.0

SDEF

Spherical deform blending

Version 2.0

type	name	Notes
index	Bone index 1	
index	Bone index 2	
float	Bone 1 weight	Bone 2 weight = 1.0 - Bone 1 weight
vec3	C	???
vec3	R0	???
vec3	R1	???

QDEF

Dual quaternion deform blending

Version 2.1

type	name	Notes
index	Bone index 1	
index	Bone index 2	
index	Bone index 3	
index	Bone index 4	
float	Bone 1 weight	Weight total not guaranteed to be 1.0
float	Bone 2 weight	Weight total not guaranteed to be 1.0
float	Bone 3 weight	Weight total not guaranteed to be 1.0
float	Bone 4 weight	Weight total not guaranteed to be 1.0

Vertex Data

The vertex section starts with an int defining how many vertices there are (Note that it is a signed int).

Name	Type	Notes
Position	vec3	XYZ
Normal	vec3	XYZ
UV	vec2	XY
Additional vec4	vec4[N]	N is defined in the Globals (Can be 0)
Weight deform type	byte	0 = BDEF1, 1 = BDEF2, 2 = BDEF4, 3 = SDEF, 4 = QDEF
Weight deform	~	See Deform Types above
Edge scale	float	Pencil-outline scale (1.0 should be around 1 pixel)

Surface

Surfaces are in groups of 3, they define a single triangle face based on the vertices that they are referencing.

[surface 0 -> A, surface 1 -> B, surface 2 -> C] = triangle [vertex A, vertex B, vertex C]

Triangle facing is defined in clockwise winding order (As with DirectX).

Surface Data

The surface section starts with an int defining how many surfaces there are (Note that it is a signed int).

Name	Type	Notes
Vertex index	index	See Index Types above

Additional Surface Types (PMX 2.1)

The material data is extended for PMX 2.1 to include point and line drawing. See the Material section to learn more about this.

Texture

This is a table of texture paths. The format of the paths is not defined, so it may be system dependant and it may feature absolute paths, this information is beyond the spec.

The file structure itself is also not defined, so textures may be in sub folders or may be expecting textures to be pre-defined by the implementation.

Some implementations will change texture behaviour based on file name, such as using "_s" as sphere map or "_n" as normal map or may have texture entries that do not even reference files but reference some other data.

There is also no preferred texture format; the most common texture formats are probably BMP, PNG, TGA but JPG and DDS formats are also sometimes used.

Reserved texture names

Textures "toon01.bmp" through to "toon10.bmp" are reserved. A model should not be referencing these textures via the texture data section. The resulting behaviour if this occurs is implementation dependant, some implementations will ignore these entries or some will directly reference internal textures.

Texture Data

The texture section starts with an int defining how many textures there are (Note that it is a signed int).

Name	Type	Notes
Path	text	File system path to a texture file (Usually relative)

Material

Internally, materials would define the shading for the model when being rendered.

PMX is intended for material-based rendering, not physically-based rendering,

Material Flags

Materials can have the following bit flags:

Bit index	Name	Set effect	Version
0	No-cull	Disables back-face culling	2.0
1	Ground shadow	Projects a shadow onto the geometry	2.0
2	Draw shadow	Renders to the shadow map	2.0
3	Receive shadow	Receives a shadow from the shadow map	2.0
4	Has edge	Has "pencil" outline	2.0
5	Vertex colour	Uses additional vec4 1 for vertex colour	2.1
6	Point drawing	Each of the 3 vertices are points	2.1
7	Line drawing	The triangle is rendered as lines	2.1

Point and Line drawing (PMX 2.1)

The point-drawing flag will override the line-drawing flag if both are set.

- A standard triangle is rendered with vertices [A->B->C].
- With point drawing each vertex is rendered as individual points [A, B, C] resulting in 3 points.

- Line drawing will render 3 lines [A->B, B->C, C->A] resulting in 3 lines.

When point rendering is enabled the edge scale value from the material (Not the vertices) will control the size of the point.

It is undefined if point or lines will have edges if the edge flag is set.

Material Data

The material section starts with an int defining how many materials there are (Note that it is a signed int).

Name	Type	Notes
Material name local	text	Handy name for the material (Usually Japanese)
Material name universal	text	Handy name for the material (Usually English)
Diffuse colour	vec4	RGBA colour (Alpha will set a semi-transparent material)
Specular colour	vec3	RGB colour of the reflected light
Specular strength	float	The "size" of the specular highlight
Ambient colour	vec3	RGB colour of the material shadow (When out of light)
Drawing flags	flag	See Material Flags
Edge colour	vec4	RGBA colour of the pencil-outline edge (Alpha for semi-transparent)
Edge scale	float	Pencil-outline scale (1.0 should be around 1 pixel)
Texture index	index	See Index Types, this is from the texture table by default
Environment index	index	Same as texture index, but for environment mapping
Environment blend mode	byte	0 = disabled, 1 = multiply, 2 = additive, 3 = additional vec4*
Toon reference	byte	0 = Texture reference, 1 = internal reference
Toon value	index/byte	Behaviour depends on Toon reference value**
Meta data	text	This is used for scripting or additional data
Surface count	int	How many surfaces this material affects***

* Environment blend mode 3 will use the first additional vec4 to map the environment texture, using just the X and Y values as the texture UV. It is mapped as an additional texture layer. This may conflict with other uses for the first additional vec4.

** Toon value will be a texture index much like the standard texture and environment texture indexes unless the Toon reference byte is equal to 1, in which case Toon value will be a byte that references a set of 10 internal toon textures (Most implementations will use "toon01.bmp" to "toon10.bmp" as the internal textures, see the reserved names for Textures above).

*** Surface count will always be a multiple of 3. It is based on the offset of the previous material through to the size of the current material. If you add up all the surface counts for all materials you should end up with the total number of surfaces.

Bone

Bones are used for vertex animation.

Bone Flags

Bones can have the following bit flags:

Bit index	Name	Set effect	Version
0	Indexed tail position	Is the tail position a vec3 or bone index	2.0
1	Rotatable	Enables rotation	2.0
2	Translatable	Enables translation (shear)	2.0
3	Is visible	???	2.0
4	Enabled	???	2.0
5	IK	Use inverse kinematics (physics)	2.0
8	Inherit rotation	Rotation inherits from another bone	2.0
9	Inherit translation	Translation inherits from another bone	2.0
10	Fixed axis	The bone's shaft is fixed in a direction	2.0
11	Local co-ordinate	???	2.0
12	Physics after deform	???	2.0
13	External parent deform	???	2.0

Inherit Bone

Name	Type	Notes
Parent index	index	See Index Types
Parent influence	float	Weight of how much influence this parent has on this bone

Bone Fixed Axis

Name	Type	Notes
Axis direction	vec3	Direction this bone points

Bone Local Co-ordinate

Name	Type	Notes
X vector	vec3	???
Z vector	vec3	???

Bone External Parent

Name	Type	Notes
Parent index	index	???

IK Angle Limit

Name	Type	Notes

Limit min	vec3	Minimum angle (radians)
Limit max	vec3	Maximum angle (radians)

IK Links

Name	Type	Notes
Bone index	index	See Index Types
Has limits	byte	When equal to 1, use angle limits
IK Angle limits	~	Used if has limits is 1. See IK Angle Limit

Bone IK

Name	Type	Notes
Target index	index	See Index Types
Loop count	int	???
Limit radian	float	???
Link count	int	How many bones this IK links with
IK links	~[N]	N is link count; can be zero. See IK Links

Bone Data

The bone section starts with an int defining how many bones there are (Note that it is a signed int).

Name	Type	Notes
Bone name local	text	Handy name for the bone (Usually Japanese)
Bone name universal	text	Handy name for the bone (Usually English)
Position	vec3	The local translation of the bone
Parent bone index	index	See Index Types
Layer	int	Deformation hierarchy
Flags	flag[2]	See Bone Flags
Tail position	vec3/index	If indexed tail position flag is set then this is a bone index
Inherit bone	~	Used if either of the inherit flags are set. See Inherit Bone
Fixed axis	~	Used if fixed axis flag is set. See Bone Fixed Axis
Local co-ordinate	~	Used if local co-ordinate flag is set. See Bone Local Co-ordinate
External parent	~	Used if external parent deform flag is set. See Bone External Parent
IK	~	Used if IK flag is set. See Bone IK

Morph

Offset Data

Offset data depends on the morph type.

Index	Type	Notes	Version
0	Group		2.0
1	Vertex		2.0
2	Bone		2.0
3	UV		2.0
4	UV ext1		2.0
5	UV ext2		2.0
6	UV ext3		2.0
7	UV ext4		2.0
8	Material		2.0
9	Flip		2.1
10	Impulse		2.1

Group

Name	Type	Notes
Morph index	index	See Index Types
Influence	float	Weight of indexed morph

Vertex

Name	Type	Notes
Vertex index	index	See Index Types
Translation	vec3	Translation to apply to vertex

Bone

Name	Type	Notes
Bone index	index	See Index Types
Translation	vec3	Translation to apply to bone
Rotation	vec4	Rotation to apply to bone

UV

Name	Type	Notes
Vertex index	index	See Index Types
Floats	vec4	What these do depends on the UV ext

Material

Name	Type	Notes

Material index	index	See Index Types. -1 is all materials
	byte	
Diffuse	vec4	
Specular	vec3	
Specularity	float	
Ambient	vec3	
Edge colour	vec4	
Edge size	float	
Texture tint	vec4	
Environment tint	vec4	
Toon tint	vec4	

Flip

Name	Type	Notes
Morph index	index	See Index Types
Influence	float	Weight of indexed morph

Impulse

Name	Type	Notes
Rigid body index	index	See Index Types
Local flag	byte	
Movement speed	vec3	
Rotation torque	vec3	

Morph Data

The morph section starts with an int defining how many morphs there are (Note that it is a signed int).

Name	Type	Notes
Morph name local	text	Handy name for the morph (Usually Japanese)
Morph name universal	text	Handy name for the morph (Usually English)
Panel type	byte	See Morph Panel Types
Morph type	byte	See Morph Types
Offset size	int	Number of morph offsets
Offset data	~[N]	N is offset count; can be zero. See Offset Data

Display Frame

Frame Type

Index	Type	Notes	Version
0	Bone		2.0
1	Morph		2.0

Bone Frame Data

Name	Type	Notes
Bone index	index	See Index Types

Morph Frame Data

Name	Type	Notes
Morph index	index	See Index Types

Frame Data

Name	Type	Notes
Frame type	byte	See Frame Type
Frame data	~	See Frame Datas

Display Data

The display frame section starts with an int defining how many frames there are (Note that it is a signed int).

Name	Type	Notes
Display name local	text	Handy name for the display (Usually Japanese)
Display name universal	text	Handy name for the display (Usually English)
Special flag	byte	0 = normal frame, 1 = special frame
Frame count	int	How many frames are in this display
Frames	~[N]	N is frame count. See Frame Data

Rigid Body

Shape Type

Index	Type	Notes	Version
0	Sphere		2.0
1	Box		2.0
2	Capsule		2.0

Physics Modes

Index	Type	Notes	Version
0	Follow bone	Rigid body sticks to bone	2.0
1	Physics	Rigid body uses gravity	2.0
2	Physics + bone	Rigid body uses gravity pivoted to bone	2.0

Rigid Body Data

The rigid body section starts with an int defining how many rigid bodies there are (Note that it is a signed int).

Name	Type	Notes
Rigid body name local	text	Handy name for the rigid body (Usually Japanese)
Rigid body name universal	text	Handy name for the rigid body (Usually English)
Related bone index	index	See Index Types
Group id	byte	Group id
Non-collision group	short	Non collision mask
Shape	byte	See Shape Type
Shape size	vec3	XYZ bounds of the shape
Shape position	vec3	XYZ shape location
Shape rotation	vec3	XYZ shape rotation angles in radians
Mass	float	
Move attenuation	float	
Rotation damping	float	
Repulsion	float	
Friction force	float	
Physics mode	byte	See Physics Modes

Joint

Joint Type

Index	Type	Notes	Version
0	Spring 6DOF		2.0
1	6DOF		2.1
2	P2P		2.1
3	ConeTwist		2.1
4	Slider		2.1
5	Hinge		2.1

Joint Data

The joint section starts with an int defining how many joints there are (Note that it is a signed int).

Name	Type	Notes
Joint name local	text	Handy name for the rigid body (Usually Japanese)
Joint name universal	text	Handy name for the rigid body (Usually English)
Type	byte	See Joint Type
Rigid body index A	index	See Index Types
Rigid body index B	index	
Position	vec3	
Rotation	vec3	Rotation angle radian
Position minimum	vec3	Minimum position value
Position maximum	vec3	Maximum position value
Rotation minimum	vec3	Minimum rotation value
Rotation maximum	vec3	Maximum rotation value
Position spring	vec3	
Rotation spring	vec3	

Soft Body

Soft bodies are based on Bullet Physics and are introduced with PMX 2.1

Shape Types

Index	Type	Notes	Version
0	TriMesh		2.1
1	Rope		2.1

Flags

Bit index	Name	Set effect	Version
0	B-Link	???	2.1
1	Cluster creation	???	2.1
2	Link crossing	???	2.1

Aerodynamics Models

Index	Type	Notes	Version
0	V_Point	???	2.1
1	V_TwoSided	???	2.1
2	V_OneSided	???	2.1

Index	Type	Notes	Version
3	F_TwoSided	???	2.1
4	F_OneSided	???	2.1

Anchor Rigid Bodies

Name	Type	Notes
Rigid body index	index	See Index Types
Vertex index	index	See Index Types
Near mode	byte	???

Vertex Pins

Name	Type	Notes
Vertex index	index	See Index Types

Soft Body Data

The soft body section starts with an int defining how many soft bodies there are (Note that it is a signed int).

Name	Type	Notes
Soft body name local	text	Handy name for the soft body (Usually Japanese)
Soft body name universal	text	Handy name for the soft body (Usually English)
Shape	byte	See Shape Types
Material index	index	See Index Types
Group	byte	Group id
No collision mask	short	Non collision mask
Flags	flag	See Flags
B-link create distance	int	???
Number of clusters	int	???
Total mass	float	
Collision margin	float	
Aerodynamics model	int	See Aerodynamics Models
Config VCF	float	Velocities correction factor (Baumgarte)
Config DP	float	Damping coefficient
Config DG	float	Drag coefficient
Config LF	float	Lift coefficient
Config PR	float	Pressure coefficient
Config VC	float	Volume conversation coefficient

Name	Type	Notes
Config DF	float	Dynamic friction coefficient
Config MT	float	Pose matching coefficient
Config CHR	float	Rigid contacts hardness
Config KHR	float	Kinetic contacts hardness
Config SHR	float	Soft contacts hardness
Config AHR	float	Anchors hardness
Cluster SRHR_CL	float	Soft vs rigid hardness
Cluster SKHR_CL	float	Soft vs kinetic hardness
Cluster SSHR_CL	float	Soft vs soft hardness
Cluster SR_SPLT_CL	float	Soft vs rigid impulse split
Cluster SK_SPLT_CL	float	Soft vs kinetic impulse split
Cluster SS_SPLT_CL	float	Soft vs soft impulse split
Interation V_IT	int	Velocities solver iterations
Interation P_IT	int	Positions solver iterations
Interation D_IT	int	Drift solver iterations
Interation C_IT	int	Cluster solver iterations
Material LST	int	Linear stiffness coefficient
Material AST	int	Area / Angular stiffness coefficient
Material VST	int	Volume stiffness coefficient
Anchor rigid body count	int	How many anchoring rigid bodies there are
Anchor rigid bodies	~[N]	N is the anchor rigid body count. See Anchor Rigid Bodies
Vertex pin count	int	How many vertex pins there are
Vertex pins	~[N]	N is the vertex pin count. See Vertex Pins

fangzhangmnm commented on May 2, 2015

where is the physics/selekton/morph information?
maybe could you give me the japanese version?