# PMX ファイルの剛体データと BepuPhysics 利用の調査

## PMX ファイルに含まれる剛体情報

PMX ファイルの「剛体」(RigidBody) セクションでは、物理シミュレーションに使う形状や剛体タイプが定義されています。C# 用 PMX パーサー PMXParser のソースを見ると、剛体は次のようなプロパティを持ちます[[1]](https://raw.githubusercontent.com/ikorin24/PMXParser/master/PMXParser/PMXObject.cs#:~:text=public%20byte%20Group%20,get%3B%20internal%20set%3B)。

| フィールド名 | 内容 |
| --- | --- |
| **Group** | 剛体が所属する衝突グループ番号 (0‑15)。同一グループ同士の衝突を禁止するためのフィルタに利用する。 |
| **GroupTarget** | 16 ビットのビットマスク。衝突させたくないグループのビットを 1 に設定する。 |
| **Shape** | 剛体の形状。列挙 RigidBodyShape には **Sphere(0)、Box(1)、Capsule(2)** が定義されている[[2]](https://raw.githubusercontent.com/ikorin24/PMXParser/master/PMXParser/PMXObject.cs#:~:text=public%20enum%20RigidBodyShape%20%3A%20byte,1%2C%20Capsule%20%3D%202%2C)。 |
| **Size** | 形状の大きさ (半径や幅/高さ/奥行き)。Sphere は (半径, 半径, 半径), Box は (幅, 高さ, 奥行き), Capsule は (半径, 高さ, 半径) のように扱う。 |
| **Position**／**RotationRadian** | 剛体の原点（モデルのボーンを基準としたローカル座標）と回転（ラジアン）。 |
| **Mass** | 剛体質量。0 に近いほど静的に扱われ、値が大きいほど慣性が大きくなる。 |
| **TranslationAttenuation**／**RotationAttenuation** | 移動・回転の減衰率。値が 0 に近いほど減衰が強く、1 に近いほど無減衰。 |
| **Recoil** | 反発係数 (バウンス)。 |
| **Friction** | 摩擦係数。 |
| **PhysicsType** | 剛体の物理タイプ。RigidBodyPhysicsType 列挙に次の 3 種類がある[[3]](https://raw.githubusercontent.com/ikorin24/PMXParser/master/PMXParser/PMXObject.cs#:~:text=public%20enum%20RigidBodyShape%20%3A%20byte,1%2C%20Capsule%20%3D%202%2C)。 |

RigidBodyPhysicsType の意味:

| 値 | 意味 | MMD での役割 |
| --- | --- | --- |
| **0 = Static** | ボーンに追従し、物理エンジンの力を受けない剛体。コライダ（足裏や体）の形状として使われる。 | ボーンの姿勢に合わせて配置され、動的な物理計算はしない。 |
| **1 = Dynamic** | 完全な動的剛体。質量や摩擦・反発値に基づき物理エンジンでシミュレーションされる。 | 髪やスカートなど自由に揺れる部分。ボーンやアニメーションには影響しない。 |
| **2 = DynamicAndBonePosition** | 動的剛体だが、剛体の位置・回転をボーンに反映する。 | 物理計算で得た位置・角度をボーンに書き戻し、ボーンを駆動する。 |

剛体のリストとは別に、PMX には **Joint** 情報があり、剛体同士をバネ付き 6DOF ジョイントなどで結ぶことができます[[4]](https://raw.githubusercontent.com/ikorin24/PMXParser/master/PMXParser/PMXObject.cs#:~:text=public%20byte%20Group%20,)。これは物理エンジン側で拘束を構成するために必要です。

## BepuPhysics の剛体種別

BepuPhysics v2 は C# 製の高速なリアルタイム物理エンジンです。剛体は BodyDescription で定義し、Simulation.Bodies.Add() でシミュレーションに追加します。Bepu では**質量 (inverse mass) と慣性テンソルの値がゼロかどうか**で剛体の性質が決まります。Q&A ドキュメントでは、外力で動かないが速度を直接変更出来る **キネマティック剛体**の作り方が次のように説明されています[[5]](https://raw.githubusercontent.com/bepu/bepuphysics2/master/Documentation/QuestionsAndAnswers.md#:~:text=,can%20still%20have%20a%20velocity)。

「外部からの衝突で動かしたくないが速度は持たせたい場合、BodyDescription.CreateKinematic を使うか、BodyDescription の逆質量・逆慣性テンソルをすべて 0 にすると良い。キネマティック剛体は無限大の質量を持ち、力で動かないが、速度を直接変更できる」[[5]](https://raw.githubusercontent.com/bepu/bepuphysics2/master/Documentation/QuestionsAndAnswers.md#:~:text=,can%20still%20have%20a%20velocity)。

逆に、**逆質量や逆慣性テンソルに 0 以外の値があれば動的剛体 (Dynamic)** になります[[6]](https://raw.githubusercontent.com/bepu/bepuphysics2/master/Documentation/QuestionsAndAnswers.md#:~:text=Any%20nonzero%20component%20in%20the,results%20in%20a%20dynamic%20body)。動的剛体は他の物体から力を受け、重力や衝突で移動・回転します。また、Bepu には速度ゼロで完全に固定された **Static** オブジェクトを表す StaticDescription 構造体があり、「シミュレーションに追加すると衝突はするが速度がなく、力で動かない」[[7]](https://raw.githubusercontent.com/bepu/bepuphysics2/master/BepuPhysics/StaticDescription.cs#:~:text=%2F%2F%2F%20,summary%3E%20public%20TypedIndex%20Shape)と説明されています。

Bepu v2 には旧バージョンにあった**反発係数 (restitution)** の設定が存在しません。「Q&A」によると「バウンス/反発の物性値は存在せず、代わりに摩擦係数とバネの周波数・減衰率・最大復帰速度を設定して反発効果を実現する」[[8]](https://raw.githubusercontent.com/bepu/bepuphysics2/master/Documentation/QuestionsAndAnswers.md#:~:text=,material%20property)とのことです。したがって、PMX の Recoil はそのままの形で割り当てられず、バネの設定を調整する必要があります。摩擦係数については INarrowPhaseCallbacks.ConfigureContactManifold で ContactMaterialProperties の FrictionCoefficient を設定できます。

## PMX 剛体と BepuPhysics の対応方法

以下に、PMX 内の剛体情報を BepuPhysics に取り込んで物理シミュレーションを行う際の対応方法をまとめます。

### 1. 剛体タイプと Bepu の剛体種別

| PMX PhysicsType | Bepu の扱い | 実装ポイント |
| --- | --- | --- |
| **Static (0)** | **キネマティックまたはスタティック剛体**。ボーンに追従し、物理計算の結果で動かさない。 | BodyDescription.CreateKinematic で質量 0 のボディを作成し、毎フレームボーンの行列から BodyReference.Pose.Position/Orientation を更新する。または完全に動かさない場合は StaticDescription を用いて Simulation.Statics.Add() する[[7]](https://raw.githubusercontent.com/bepu/bepuphysics2/master/BepuPhysics/StaticDescription.cs#:~:text=%2F%2F%2F%20,summary%3E%20public%20TypedIndex%20Shape)。 |
| **Dynamic (1)** | **動的剛体**。質量と慣性を持ち、物理エンジンで自由に動く。 | PMX の Mass を利用して形状 (Sphere/Box/Capsule) から ComputeInertia(mass) を計算し、BodyDescription.CreateDynamic で生成する。初期位置・回転は Position と RotationRadian を参照。ボーンへの書き戻しは行わない。 |
| **DynamicAndBonePosition (2)** | **動的剛体 + ボーン同期**。動的にシミュレーションし、結果の位置・回転を対応するボーンに反映する。 | CreateDynamic で動的剛体を作成し、毎フレームシミュレーション後に BodyReference.Pose からボーンのローカル変換を更新する。これによりボーンが物理で動く。 |

### 2. 形状の作成

PMX の Shape 列挙[[2]](https://raw.githubusercontent.com/ikorin24/PMXParser/master/PMXParser/PMXObject.cs#:~:text=public%20enum%20RigidBodyShape%20%3A%20byte,1%2C%20Capsule%20%3D%202%2C)に従い、BepuPhysics で対応する形状を生成します。

| PMX 形状 | Bepu の形状クラス | サイズ変換 |
| --- | --- | --- |
| **Sphere** | Sphere | Sphere(radius)。PMX の Size.X を半径として使う。 |
| **Box** | Box | Box(width, height, depth)。PMX の Size をそのまま渡す。 |
| **Capsule** | Capsule | Capsule(radius, length)。半径は Size.X、高さは Size.Y。Bepu では全長に相当するため、中心球を除いた長さを与える。 |

形状は TypedIndex shapeIndex = Simulation.Shapes.Add(shape); でシミュレーションに登録し、CollidableDescription に渡します。

### 3. 質量・慣性の設定

動的剛体の場合、Bepu は形状から慣性テンソルを計算できます。例えば球の場合:

var shape = new Sphere(radius);  
var inertia = shape.ComputeInertia(pmxRigidBody.Mass);  
var collidable = new CollidableDescription(shapeIndex, speculativeMargin: BodyDescription.GetDefaultSpeculativeMargin(shape));  
var bodyDescription = BodyDescription.CreateDynamic(new RigidPose(position, orientation), inertia, collidable, BodyDescription.GetDefaultActivity(shape));

キネマティック剛体の場合は CreateKinematic を使い、質量や慣性テンソルを設定しません[[9]](https://raw.githubusercontent.com/bepu/bepuphysics2/master/BepuPhysics/BodyDescription.cs#:~:text=public%20static%20BodyDescription%20CreateKinematic,)。静的地形の場合は StaticDescription に形状と姿勢を設定して Simulation.Statics.Add() します[[7]](https://raw.githubusercontent.com/bepu/bepuphysics2/master/BepuPhysics/StaticDescription.cs#:~:text=%2F%2F%2F%20,summary%3E%20public%20TypedIndex%20Shape)。

### 4. 減衰・摩擦・反発の扱い

PMX の TranslationAttenuation と RotationAttenuation は減衰率です。Bepu には剛体ごとの減衰パラメータが存在しないため、IPoseIntegratorCallbacks の実装内で速度に手動で減衰を掛けるのが一般的です。たとえば Substep の最後で body.LinearVelocity \*= (1 - translationAttenuation); のようにすることで減衰を再現できます。

PMX の Friction は Bepu の ContactMaterialProperties.FrictionCoefficient に相当します。INarrowPhaseCallbacks.ConfigureContactManifold で manifold.FrictionCoefficient = pmxRigidBody.Friction; のように設定します。

Recoil (反発係数) は Bepu v2 では直接指定できません。ドキュメントにある通り「バウンス/反発の物性値は存在せず、代わりに摩擦とバネの周波数・減衰率・最大復帰速度を使う」[[8]](https://raw.githubusercontent.com/bepu/bepuphysics2/master/Documentation/QuestionsAndAnswers.md#:~:text=,material%20property)ため、反発値に応じて SpringSettings.Frequency や MaximumRecoveryVelocity を調整し、バウンスの強さを再現します。

### 5. 衝突グループのフィルタリング

PMX の Group と GroupTarget は、どの剛体同士が衝突するかを指定するためのビットマスクです。Bepu ではコリジョンフィルタリングを行うために INarrowPhaseCallbacks.AllowContactGeneration を実装します。剛体ごとにグループ番号とマスクを保持し、2 つの剛体の (groupA & groupMaskB) != 0 && (groupB & groupMaskA) != 0 を満たすときのみ衝突を許可するようにします。グループ情報は RigidBody.Tag や独自辞書に保存してコールバック内で参照します。

### 6. ジョイントの構成

PMX のジョイントはバネ付き 6DOF ジョイントが主で、TranslationMinLimit/MaxLimit と RotationRadianMinLimit/MaxLimit によって移動・回転の制限範囲が定義されています[[4]](https://raw.githubusercontent.com/ikorin24/PMXParser/master/PMXParser/PMXObject.cs#:~:text=public%20byte%20Group%20,)。Bepu では以下のような拘束を組み合わせて近似します。

1. **ボールソケット (BallSocket)** – 剛体間の回転を自由にし、位置を一致させる拘束。
2. **TwistLimit／SwingLimit** – 回転の範囲を制限する。PMX の回転制限範囲を SpringSettings として指定し、ばねの強さを調整して柔らかい拘束にできます。
3. **LinearAxisMotor or DistanceLimit** – 移動の範囲を制限する。PMX の TranslationMinLimit から TranslationMaxLimit までを許容範囲に設定し、中心位置にはばね (SpringSettings) を加えることでスプリングジョイントを実現。

各ジョイントは Simulation.Solver.Add で登録します。PMX の RotationSpring／TranslationSpring は Bepu の SpringSettings (周波数と減衰比) に対応させ、剛体を柔らかく接続します。

### 7. フレーム更新とボーンとの同期

1. PMX を読み込んだ後、全剛体を Bepu に登録する。
2. 描画フレームごとに、Type 0 の剛体はボーンの行列を参照して BodyReference.Pose を更新する (Bepu で言うキネマティック更新)。
3. Simulation.Timestep(dt) で物理シミュレーションを進める。
4. Type 2 の剛体については、シミュレーション結果の BodyReference.Pose をボーンのローカル位置・回転に書き戻す。これにより物理計算がスケルトンに反映される。
5. OpenTK で描画する際には、シミュレーション後のボーン行列でスキニングを行う。

## まとめ

PMX ファイルの剛体情報は、**形状 (Sphere / Box / Capsule)、剛体タイプ (Static / Dynamic / DynamicAndBonePosition)、質量、減衰、反発、摩擦、衝突グループ**などを持ちます[[1]](https://raw.githubusercontent.com/ikorin24/PMXParser/master/PMXParser/PMXObject.cs#:~:text=public%20byte%20Group%20,get%3B%20internal%20set%3B)。これらは C# で BepuPhysics にマッピングできます。BepuPhysics では

* 逆質量・逆慣性テンソルが 0 のものがキネマティック剛体となり、外力では動かないが速度を持てる[[5]](https://raw.githubusercontent.com/bepu/bepuphysics2/master/Documentation/QuestionsAndAnswers.md#:~:text=,can%20still%20have%20a%20velocity)。
* 逆質量が非ゼロのものは動的剛体として完全に物理シミュレーションされる[[6]](https://raw.githubusercontent.com/bepu/bepuphysics2/master/Documentation/QuestionsAndAnswers.md#:~:text=Any%20nonzero%20component%20in%20the,results%20in%20a%20dynamic%20body)。
* 完全に静的な衝突形状は StaticDescription で追加し、全く動かさない[[7]](https://raw.githubusercontent.com/bepu/bepuphysics2/master/BepuPhysics/StaticDescription.cs#:~:text=%2F%2F%2F%20,summary%3E%20public%20TypedIndex%20Shape)。

Bepu v2 には **反発係数** という概念がなく、バネの周波数・減衰率と最大復帰速度でバウンスを再現する[[8]](https://raw.githubusercontent.com/bepu/bepuphysics2/master/Documentation/QuestionsAndAnswers.md#:~:text=,material%20property)点に注意が必要です。また、剛体同士の衝突制御は INarrowPhaseCallbacks を実装して行います。これらのマッピングを適切に行えば、PMX モデルの剛体情報を BepuPhysics と OpenTK で活かした物理表現が可能になります。

## physics.txt に基づく追加調査と具体的な利用方法

ユーザー提供の PMX モデル「艾蓮.pmx」には、付随する physics.txt があり、PMX 内の物理設定を人間が読みやすいテキストとして出力したものです。ファイル冒頭には重力や減衰係数などのグローバル設定が記載され、続いて数十個の剛体が RigidBody: 行として列挙されています。例えば次の行では「頭」という剛体がボーン 10 に紐付けられ、質量 14.625、カプセル形状、摩擦 0.5、ボーン追従モード (Mode:FollowBone) として定義されていることが分かります[[10]](file:///home/oai/share/physics.txt)。

RigidBody: 頭 BoneIndex:10 Mass:14.625 Shape:Capsule … Friction:0.5 Position:〈0, 18.080936, −0.17313504〉 … Mode:FollowBone[[10]](file:///home/oai/share/physics.txt)

この形式で首や胴体、髪などの剛体が続き、それぞれの質量・形状・サイズ・減衰係数・摩擦・反発係数・衝突グループ・マスク・モードが記録されています。physics.txt の後半には Joint: 行が並び、剛体同士 (BodyA と BodyB) をどのような範囲で拘束するかが PosMin/PosMax、RotMin/RotMax（平行移動・回転の許容範囲）および SpringPos/SpringRot（ばね定数）として示されています。

### physics.txt のデータを BepuPhysics に取り込む方法

1. **グローバル設定** – Gravity は Simulation の重力ベクトルとしてそのまま利用します。EffectiveGravity が別途定義されていれば、これは物理演算に適用する重力の実効値なので同じく重力ベクトルに設定します。SolverIterationCount や SubstepCount は Simulation 生成時にイテレーション数として渡し、Damping はシミュレーション後に速度に係数を掛ける処理で再現します。BoneBlendFactor は Type2 の剛体で、ボーンアニメーションと物理のブレンド比率として用います。 LockTranslation が True の場合は大地の高さ (GroundHeight) から下方向の移動を禁止するなどの制約を付加します。
2. **剛体生成** – 各 RigidBody: 行から名称・ボーンインデックス・質量 (Mass)・形状 (Shape)・サイズ (Size)・減衰 (LinearDamping/AngularDamping)・摩擦 (Friction)・反発 (Restitution)・グループ (Group)・マスク (Mask)・モード (Mode) を読み取り、BepuPhysics の剛体を作成します。
3. Shape は「Sphere」「Box」「Capsule」の3種類であり、それぞれ Sphere(radius)、Box(width,height,depth)、Capsule(radius,length) に変換します。カプセルの長さは PMX の Size.Y であり、Bepu では中心球を除いた筒の長さを与えることに注意します。
4. 質量が0、または Mode が FollowBone (PMX では Type0) の剛体はボーン追従用であり、BodyDescription.CreateKinematic や StaticDescription でキネマティック/静的ボディを作成します[[7]](https://raw.githubusercontent.com/bepu/bepuphysics2/master/BepuPhysics/StaticDescription.cs#:~:text=%2F%2F%2F%20,summary%3E%20public%20TypedIndex%20Shape)[[9]](https://raw.githubusercontent.com/bepu/bepuphysics2/master/BepuPhysics/BodyDescription.cs#:~:text=public%20static%20BodyDescription%20CreateKinematic,)。これらは毎フレームボーン姿勢に基づいて Pose を更新します。
5. Mode が Physics (Type1) の剛体は純粋な動的ボディとして CreateDynamic を使用し、PMX の Mass から慣性テンソルを計算して登録します。Mode が PhysicsAndBone (Type2) の剛体は動的にシミュレーションし、結果の Pose をボーンへ書き戻します。
6. LinearDamping と AngularDamping は剛体固有の速度減衰係数として保存し、IPoseIntegratorCallbacks で速度に乗算する方式で適用します。
7. Friction は ContactMaterialProperties.FrictionCoefficient に割り当てます。Bepu v2 には Restitution が無いため、反発効果はバネ定数や最大復帰速度で近似します[[8]](https://raw.githubusercontent.com/bepu/bepuphysics2/master/Documentation/QuestionsAndAnswers.md#:~:text=,material%20property)。
8. Group と Mask は衝突フィルタのビットマスクとして格納し、INarrowPhaseCallbacks.AllowContactGeneration で (groupA & maskB) != 0 && (groupB & maskA) != 0 の場合のみ衝突を許可します。
9. **ジョイント生成** – Joint: 行から剛体 A/B の名前を基にそれぞれの BodyHandle を取得し、Bepu の拘束を組み立てます。
10. PosMin/PosMax が (0,0,0) で移動範囲がない場合は BallSocket を使って２つの剛体のポイントを一致させます。範囲がある場合は DistanceLimit や LinearAxisLimit を用いて最小・最大距離を設定します。
11. RotMin/RotMax は各軸の回転許容範囲であり、X 軸回転は TwistLimit、Y・Z 軸回転は SwingLimit で制約します。ラジアン単位であることを忘れないようにします。
12. SpringPos/SpringRot はばねの硬さを示し、Bepu の SpringSettings（自然振動数と減衰率）へ変換します。大きな値は強いバネを表し、0 はばねなしに相当します。
13. ジョイントの作成後は Simulation.Solver.Add() に登録してシミュレーションに組み込みます。
14. **フレーム更新の流れ** – アニメーションと物理演算を組み合わせる場合、以下のステップで更新します。
15. すべての Type0 (ボーン追従) 剛体について、対応するボーンのローカル行列から Pose.Position と Pose.Orientation を更新します。
16. Bepu の Simulation.Timestep(deltaTime) を呼んで物理シミュレーションを進めます。
17. Type2 (PhysicsAndBone) の剛体については、シミュレーション結果の BodyReference.Pose をボーンのローカル変換に書き戻します。このとき BoneBlendFactor を使ってアニメーションと物理の結果を線形補間することができます。
18. OpenTK でモデルを描画するときは、更新されたボーン行列でスキニングを行います。

以上の手順で physics.txt の情報を正しく読み取り、BepuPhysics の各 API にマッピングすることで、C# と OpenTK(GL ES) を用いて MikuMikuDance と同様の物理挙動を再現できます。

[[1]](https://raw.githubusercontent.com/ikorin24/PMXParser/master/PMXParser/PMXObject.cs#:~:text=public%20byte%20Group%20,get%3B%20internal%20set%3B) [[2]](https://raw.githubusercontent.com/ikorin24/PMXParser/master/PMXParser/PMXObject.cs#:~:text=public%20enum%20RigidBodyShape%20%3A%20byte,1%2C%20Capsule%20%3D%202%2C) [[3]](https://raw.githubusercontent.com/ikorin24/PMXParser/master/PMXParser/PMXObject.cs#:~:text=public%20enum%20RigidBodyShape%20%3A%20byte,1%2C%20Capsule%20%3D%202%2C) [[4]](https://raw.githubusercontent.com/ikorin24/PMXParser/master/PMXParser/PMXObject.cs#:~:text=public%20byte%20Group%20,) raw.githubusercontent.com

<https://raw.githubusercontent.com/ikorin24/PMXParser/master/PMXParser/PMXObject.cs>

[[5]](https://raw.githubusercontent.com/bepu/bepuphysics2/master/Documentation/QuestionsAndAnswers.md#:~:text=,can%20still%20have%20a%20velocity) [[6]](https://raw.githubusercontent.com/bepu/bepuphysics2/master/Documentation/QuestionsAndAnswers.md#:~:text=Any%20nonzero%20component%20in%20the,results%20in%20a%20dynamic%20body) [[8]](https://raw.githubusercontent.com/bepu/bepuphysics2/master/Documentation/QuestionsAndAnswers.md#:~:text=,material%20property) raw.githubusercontent.com

<https://raw.githubusercontent.com/bepu/bepuphysics2/master/Documentation/QuestionsAndAnswers.md>

[[7]](https://raw.githubusercontent.com/bepu/bepuphysics2/master/BepuPhysics/StaticDescription.cs#:~:text=%2F%2F%2F%20,summary%3E%20public%20TypedIndex%20Shape) raw.githubusercontent.com

<https://raw.githubusercontent.com/bepu/bepuphysics2/master/BepuPhysics/StaticDescription.cs>

[[9]](https://raw.githubusercontent.com/bepu/bepuphysics2/master/BepuPhysics/BodyDescription.cs#:~:text=public%20static%20BodyDescription%20CreateKinematic,) raw.githubusercontent.com

<https://raw.githubusercontent.com/bepu/bepuphysics2/master/BepuPhysics/BodyDescription.cs>

[[10]](file:///home/oai/share/physics.txt) physics.txt

<file:///home/oai/share/physics.txt>