マルチ解像度＋アニメーション完全例

```
json
```

```
{
  "$schema": "https://mrpaf.org/schemas/v2.0.1/mrpaf.schema.json",
  "format": "MRPAF",
  "version": "2.0.1",
  "metadata": {
    "title": "アニメーションキャラクター",
    "author": "PixelArtist",
    "created": "2024-01-15T10:00:00Z",
    "tags": ["character", "rpg", "animated"]
  },
  "canvas": {
    "baseWidth": 32,
    "baseHeight": 32,
    "pixelUnit": 1.0,
    "backgroundColor": "#00000000"
  },
  "coordinateSystem": {
    "origin": "top-left",
    "xAxis": "right",
    "yAxis": "down",
    "unit": "subpixel",
    "baseUnit": 1.0,
    "subPixelPrecision": 4
  },
  "compressionProfile": {
    "name": "balanced",
    "settings": {
      "autoSelect": true,
      "quality": 0.95
    }
  },
  "palette": [
    {"id": 0, "name": "透明", "hex": "#00000000"},
    {"id": 1, "name": "肌", "hex": "#FDBCB4"},
    {"id": 2, "name": "髪", "hex": "#8B4513"},
    {"id": 3, "name": "服", "hex": "#4169E1"},
    {"id": 4, "name": "目", "hex": "#000000"},
    {"id": 5, "name": "白目", "hex": "#FFFFFF"}
  ],
  "layers": [
    {
      "id": 0,
      "name": "ベースボディ",
      "resolution": {
        "pixelArraySize": {"width": 32, "height": 32},
        "scale": 1.0,
```

```
      "effectiveSize": {"width": 32, "height": 32}
    },
    "placement": {
      "x": 0,
      "y": 0
    },
    "pixels": {
      "encoding": "rle",
      "data": "0:32|0:8,1:16,0:8|..."
    }
  },
  {
    "id": 1,
    "name": "顔ディテール",
    "resolution": {
      "pixelArraySize": {"width": 32, "height": 16},
      "scale": 4.0
      // effectiveSizeは自動計算されるため記載しない
    },
    "placement": {
      "x": 12,
      "y": 8,
      "allowSubPixel": true
    },
    "pixels": {
      "encoding": "sparse",
      "dimensions": {"width": 32, "height": 16},
      "defaultValue": 0,
      "data": [
        {"x": 8, "y": 4, "color": 4},
        {"x": 9, "y": 4, "color": 5},
        {"x": 24, "y": 4, "color": 4},
        {"x": 25, "y": 4, "color": 5}
      ]
    }
  }
],
"animations": {
  "idle": {
    "fps": 2,
    "loops": true,
    "frames": [
      {
        "duration": 500,
        "layers": [0, 1]
      },
      {
```

```
        "duration": 100,
        "layers": [0],
        "overrides": {
          "1": {"visible": false}
        }
      }
    ],
    "tags": ["character", "idle"]
  },
  "walk": {
    "fps": 8,
    "loops": true,
    "frames": [
      {
        "duration": 125,
        "layers": [0, 1],
        "tweens": {
          "0": {
            "placement": {
              "y": {"from": 0, "to": -0.5, "easing": "ease-out"}
            }
          }
        },
        "events": [
          {"type": "sound", "resourceId": "footstep", "time": 0}
        ]
      },
      {
        "duration": 125,
        "layers": [0, 1],
        "tweens": {
          "0": {
            "placement": {
              "y": {"from": -0.5, "to": 0, "easing": "ease-in"}
            }
          }
        }
      }
    ],
    "tags": ["character", "movement"]
  }
},
"animationController": {
  "defaultAnimation": "idle",
  "transitions": [
    {
      "from": "idle",
```

```
        "to": "walk",
        "condition": "moving",
        "duration": 100
      }
    ]
  },
  "resources": {
    "sounds": {
      "footstep": {
        "uri": "./sounds/step.ogg",
        "type": "audio/ogg"
      }
    }
  }
}
```# Multi-Resolution Pixel Art Format (MRPAF) Specification v2.0.1

目次

1. [はじめに](#はじめに)
2. [背景と目的](#背景と目的)
3. [MRPAFの特徴と優位性](#mrpafの特徴と優位性)
4. [ユースケース](#ユースケース)
5. [技術仕様](#技術仕様)
6. [実装ガイド](#実装ガイド)
7. [ツールとエコシステム](#ツールとエコシステム)
8. [ライセンスと貢献](#ライセンスと貢献)

はじめに

Multi-Resolution Pixel Art Format (MRPAF - マーパフ) は、現代のゲーム開発やデジタルアート制作の

MRPAFとは

MRPAFは、以下の要素を統合的に管理するJSONベースのフォーマットです：

- **マルチ解像度レイヤー**: 1つの作品内で異なる解像度を混在
- **アニメーションデータ**: フレーム、トゥイーン、イベント
- **メタデータ**: 作品情報、ライセンス、制作履歴
- **リソース管理**: 効果音、参照画像、スクリプト

誰のためのフォーマットか

- **ゲーム開発者**: 効率的なアセット管理とパフォーマンス最適化
- **ピクセルアーティスト**: 詳細な表現と作業効率の両立
- **ツール開発者**: 標準化されたフォーマットでの相互運用性
- **コンテンツクリエイター**: アニメーションとインタラクティブ要素の統合

背景と目的

既存フォーマットの課題

従来のドット絵フォーマットには以下の制限がありました：

1. **単一解像度の制約**
   - PNG/GIF: 全体が同じ解像度
   - Aseprite: レイヤーは同一サイズ
   - 結果: 詳細部分のために全体が高解像度になり、ファイルサイズが肥大化

2. **アニメーション表現の限界**
   - 整数座標のみのフレーム切り替え
   - 滑らかな動きの表現が困難
   - 補間やイージングの標準サポートなし

3. **メタデータの分散**
   - 画像データと制作情報が分離
   - バージョン管理の困難さ
   - ライセンス情報の欠如

MRPAFの開発目的

1. **効率性**: 必要な部分だけ高解像度にすることでファイルサイズを最適化
2. **表現力**: サブピクセル精度とマルチ解像度による豊かな表現
3. **統合性**: 画像、アニメーション、メタデータを1つのファイルで管理
4. **互換性**: 既存ツールからの移行パスと標準化された仕様
5. **拡張性**: 将来の技術進化に対応できる柔軟な構造

MRPAFの特徴と優位性

1. マルチ解像度レイヤーシステム

仕組み

[ベースレイヤー: 32x32]
├── [キャラクター本体: 1倍解像度]
├── [顔の詳細: 4倍解像度 (実効8x8領域)]
└── [装飾品: 2倍解像度 (実効16x16領域)]

利点
- **メモリ効率**: 重要な部分のみ高解像度
- **描画パフォーマンス**: GPUの階層的レンダリングに最適
- **編集の柔軟性**: レイヤーごとに独立した解像度で作業

実例
32x32のキャラクターで顔だけ4倍解像度にした場合：
- 従来方式: 128x128 = 16,384ピクセル
- MRPAF: 32x32 + 16x16 = 1,280ピクセル（約92%削減）

2. サブピクセル精度アニメーション

従来の問題

Frame1: x=10
Frame2: x=11 // 1ピクセルジャンプ

MRPAFの解決
```json
{
  "tweens": {
    "0": {
      "placement": {
        "x": {"from": 10.0, "to": 10.25, "easing": "ease-out"}
      }
    }
  }
}
```

- 0.25ピクセル単位の滑らかな移動
- カスタムイージング関数
- 60fps以上の高フレームレート対応

3. 高度な圧縮システム

自動圧縮選択

レイヤーの特性を分析して最適な圧縮方式を自動選択：

レイヤータイプ	推奨圧縮	圧縮率
高解像度・スパース	sparse	80-95%
少色数	indexed-rle	50-70%
グラデーション	base64+zlib	40-60%

4. 統合的なアセット管理

```json
{
  "layers": [...],      // ビジュアルデータ
  "animations": {...},  // モーションデータ
  "resources": {...},   // 関連アセット
  "metadata": {...}     // 制作情報
}
```

1つのファイルで完結した管理が可能。

5. 既存ツールとの高い互換性

- Asepriteからの直接エクスポート
- Unityへの自動インポート
- Web標準技術での表示（Canvas/WebGL）

ユースケース

ゲーム開発

インディーゲーム

- **課題**: 限られたリソースで高品質なビジュアル
- **解決**: 重要キャラクターの顔や手だけ高解像度
- **効果**: メモリ使用量50%削減、表現力は維持

モバイルゲーム

- **課題**: デバイスの性能とストレージ制約
- **解決**: 動的な解像度調整とストリーミング
- **効果**: 低スペック端末でも滑らかな動作

Webアプリケーション

ピクセルアートエディタ

- **活用**: プレビューと編集で異なる解像度

- **利点**: リアルタイムプレビューの高速化

NFTアート

- **活用**: オンチェーンメタデータとの統合
- **利点**: 作品の真正性と履歴の保証

教育・研究

アニメーション教材

- **活用**: フレーム補間の仕組みを視覚的に学習
- **利点**: インタラクティブな教材作成

技術仕様

バージョニング

- **現行バージョン**: 2.0.1
- **セマンティックバージョニング**: MAJOR.MINOR.PATCH
- **互換性ポリシー**:
  - MAJOR: 後方互換性のない変更
  - MINOR: 後方互換性のある機能追加
  - PATCH: バグ修正

基本構造

```
json
```

```json
{
  "$schema": "https://mrpaf.org/schemas/v2.0.1/mrpaf.schema.json",
  "format": "MRPAF",
  "version": "2.0.1",
  "metadata": {},
  "canvas": {},
  "coordinateSystem": {},
  "colorSpace": {},
  "palette": [],
  "layers": [],
  "animations": {},
  "resources": {}
}
```

JSON Schema 抜粋

主要な必須フィールドと型制約：

json

```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "type": "object",
  "required": ["format", "version", "metadata", "canvas", "palette", "layers"],
  "properties": {
    "format": {
      "type": "string",
      "const": "MRPAF"
    },
    "version": {
      "type": "string",
      "pattern": "^\\d+\\.\\d+\\.\\d+$"
    },
    "metadata": {
      "type": "object",
      "required": ["title"],
      "properties": {
        "title": {"type": "string", "minLength": 1}
      }
    },
    "coordinateSystem": {
      "type": "object",
      "properties": {
        "origin": {
          "type": "string",
          "enum": ["top-left", "bottom-left", "center"]
        },
        "unit": {
          "type": "string",
          "enum": ["pixel", "subpixel"],
          "description": "pixel: 整数座標のみ, subpixel: 浮動小数点座標許可"
        }
      }
    },
    "layers": {
      "type": "array",
      "minItems": 1,
      "items": {
        "type": "object",
        "required": ["id", "name", "resolution", "placement", "pixels"]
      }
    }
  }
}
```

公式JSON Schema

完全な検証用スキーマは以下で提供:

- https://mrpaf.org/schemas/v2.0.1/mrpaf.schema.json
- https://github.com/mrpaf/specification/schemas/

詳細仕様

1. ルートオブジェクト

フィールド	型	必須	説明
$schema	string	×	JSON Schemaの参照URL
format	string	✓	フォーマット識別子。常に "MRPAF"
version	string	✓	フォーマットバージョン（例: "2.0.1"）
metadata	object	✓	作品情報
canvas	object	✓	キャンバス設定
coordinateSystem	object	×	座標系の詳細設定
colorSpace	object	×	カラースペース設定
palette	array	✓	カラーパレット
layers	array	✓	レイヤー配列
animations	object	×	アニメーション定義
resources	object	×	外部リソース参照

2. メタデータ (metadata)

```json
{
  "metadata": {
    "title": "作品タイトル",
    "author": "作者名",
    "created": "2024-01-15T10:30:00Z",
    "modified": "2024-01-20T15:45:00Z",
    "description": "作品の説明",
    "tags": ["タグ1", "タグ2"],
    "license": "CC BY-SA 4.0",
    "work": {
      "series": "シリーズ名",
      "character": "キャラクター名",
      "scene": "シーン説明",
      "variation": "バリエーション"
    },
    "tool": {
      "name": "作成ツール名",
      "version": "1.0.0",
      "exporter": "MRPAF Exporter v1.0"
    },
    "compatibility": {
      "minVersion": "2.0.0",
      "features": ["multiResolution", "subPixel"]
    }
  }
}
```

3. キャンバス設定 (canvas)

```json
{
  "canvas": {
    "baseWidth": 24,
    "baseHeight": 24,
    "pixelUnit": 1.0,
    "backgroundColor": "#00000000",
    "pixelAspectRatio": 1.0
  }
}
```

4. 座標系 (coordinateSystem)

```json
{
  "coordinateSystem": {
    "origin": "top-left",
    "xAxis": "right",
    "yAxis": "down",
    "unit": "pixel",
    "baseUnit": 1.0,
    "subPixelPrecision": 4,
    "allowFloatingPoint": false
  }
}
```

フィールド	型	必須	説明	デフォルト値
origin	string	×	原点位置	"top-left"
xAxis	string	×	X軸の正方向	"right"
yAxis	string	×	Y軸の正方向	"down"
unit	string	×	座標単位	"pixel"
baseUnit	number	×	座標系の基準単位	1.0
subPixelPrecision	integer	×	サブピクセル精度（unitが"subpixel"時のみ有効）	4
allowFloatingPoint	boolean	×	浮動小数点座標の許可	unit依存※

※ allowFloatingPointのデフォルト値:

- `unit: "pixel"` の場合: `false`
- `unit: "subpixel"` の場合: `true`

座標単位の動作

- **unit: "pixel"**: 整数座標のみ許可。`allowFloatingPoint`の値は無視され、常に`false`として扱われます
- **unit: "subpixel"**: 浮動小数点座標を許可。`allowFloatingPoint`の値が有効で、`subPixelPrecision`で精度を指定

allowFloatingPointの詳細動作

unit値	allowFloatingPoint	実際の動作
"pixel"	true	無視（false扱い）
"pixel"	false	false
"subpixel"	true	true
"subpixel"	false	false

省略時の完全なデフォルト値

```json
{
  "coordinateSystem": {
    "origin": "top-left",
    "xAxis": "right",
    "yAxis": "down",
    "unit": "pixel",
    "baseUnit": 1.0,
    "subPixelPrecision": 4,
    "allowFloatingPoint": false
  }
}
```

canvas.pixelUnit と coordinateSystem.baseUnit の違い

- **canvas.pixelUnit**: キャンバス上の1ピクセルの物理的なサイズ（描画時のスケール）
- **coordinateSystem.baseUnit**: 座標計算時の基準単位（論理的な座標系）

例：pixelUnit=2.0、baseUnit=1.0 の場合、座標(1,1)は画面上で(2,2)の位置に描画されます。

5. カラースペース (colorSpace)

```json
{
  "colorSpace": {
    "profile": "sRGB",
    "bitDepth": 8,
    "gamma": 2.2,
    "whitePoint": "D65",
    "renderingIntent": "perceptual"
  }
}
```

フィールド	型	説明	値
profile	string	カラープロファイル	"sRGB", "AdobeRGB", "DisplayP3", "custom"
bitDepth	integer	ビット深度	8, 16, 32
gamma	number	ガンマ値	
whitePoint	string	ホワイトポイント	"D65", "D50"

6. パレット (palette)

json

```
{
  "palette": [
    {
      "id": 0,
      "name": "透明",
      "hex": "#00000000",
      "rgb": [0, 0, 0, 0],
      "hsv": [0, 0, 0, 0],
      "lab": [0, 0, 0, 0],
      "usage": "background",
      "locked": false
    }
  ]
}
```

7. レイヤー (layers)

```
json
```

```
{
  "layers": [
    {
      "id": 0,
      "name": "レイヤー名",
      "type": "raster",
      "parent": null,
      "visible": true,
      "locked": false,
      "opacity": 1.0,
      "blending": {
        "mode": "normal",
        "resolution": "target",
        "interpolation": "nearest",
        "antialiasing": false
      },
      "resolution": {
        "pixelArraySize": {
          "width": 48,
          "height": 48
        },
        "scale": 2.0,
        "effectiveSize": {
          "width": 24,
          "height": 24
        }
      },
      "placement": {
        "x": 12.0,
        "y": 12.0,
        "width": 12.0,
        "height": 12.0,
        "unit": "base",
        "allowSubPixel": true,
        "anchor": "top-left"
      },
      "transform": {
        "rotation": 0,
        "scaleX": 1.0,
        "scaleY": 1.0,
        "skewX": 0,
        "skewY": 0
      },
      "pixels": {
        "encoding": "sparse",
        "compression": "auto"
```

```
        }
      }
    ]
  }
```

8. アニメーション (animations)

```json
{
  "animations": {
    "walk": {
      "fps": 12,
      "duration": 1000,
      "loops": true,
      "pingPong": false,
      "interpolation": {
        "spatial": "linear",
        "temporal": "step",
        "easing": "linear"
      },
      "frames": [
        {
          "duration": 83.33,
          "layers": [0, 1],
          "tweens": {
            "1": {
              "placement": {
                "x": {"from": 12.0, "to": 12.5, "easing": "ease-out"},
                "y": {"from": 11.0, "to": 11.0}
              }
            }
          },
          "events": [
            {"type": "sound", "resourceId": "step", "time": 0}
          ]
        }
      ],
      "tags": ["character", "movement"],
      "priority": 1,
      "blendMode": "replace"
    },
    "idle": {
      "fps": 4,
      "loops": true,
      "frames": [...],
      "tags": ["character", "idle"],
      "priority": 0
    }
  }
}
```

アニメーション管理とタグシステム

複数アニメーション間の切り替えと管理：

```json
{
  "animationController": {
    "defaultAnimation": "idle",
    "transitions": [
      {
        "from": "idle",
        "to": "walk",
        "condition": "moving",
        "duration": 100,
        "interpolation": "ease-in-out"
      },
      {
        "from": "walk",
        "to": "jump",
        "condition": "jumping",
        "duration": 50,
        "interpolation": "ease-out"
      }
    ],
    "tagGroups": {
      "movement": ["idle", "walk", "run", "jump"],
      "combat": ["attack", "defend", "hurt"]
    }
  }
}
```

タイムライン定義（プレビュー用）

```json
{
  "timeline": {
    "duration": 5000,  // 総時間（ミリ秒）
    "tracks": [
      {
        "animation": "idle",
        "start": 0,      // 開始時刻（ミリ秒）
        "end": 1000      // 終了時刻（ミリ秒）
      },
      {
        "animation": "walk",
        "start": 1000,
        "end": 3000
      },
      {
        "animation": "jump",
        "start": 3000,
        "end": 4000
      }
    ]
  }
}
```

注意: `start`と`end`は時刻（ミリ秒）であり、フレームインデックスではありません。

イベントシステム

```json
{
  "events": [
    {
      "type": "sound",
      "resourceId": "footstep",  // リソースIDで参照
      "time": 0,                 // フレーム開始からのミリ秒
      "volume": 0.8
    },
    {
      "type": "function",
      "name": "shake",
      "time": 50,
      "parameters": {"intensity": 0.5}
    }
  ]
}
```

イベントタイミング:

- `time`: フレーム開始からの相対時間（ミリ秒）
- 0 ≤ time < duration の範囲内である必要があります
- `duration`を超える`time`値はエラー

イージング関数

名前	説明
linear	線形補間
ease-in	ゆっくり開始
ease-out	ゆっくり終了
ease-in-out	ゆっくり開始・終了
cubic-bezier(x1,y1,x2,y2)	カスタムベジェ

9. リソース (resources)

```json
{
  "resources": {
    "images": {...},
    "sounds": {...},
    "scripts": {...},
    "fonts": {...},
    "shaders": {...},
    "x-customType": {...}  // カスタムリソースタイプ
  }
}
```

リソース定義

```json
{
  "resources": {
    "sounds": {
      "footstep": {  // カテゴリ内で一意のID
        "uri": "./sounds/step.ogg",
        "type": "audio/ogg",
        "checksum": "sha256:abc123..."
      }
    },
    "images": {
      "footstep": {  // 別カテゴリなので同じIDでもOK
        "uri": "./images/footstep.png",
        "type": "image/png"
      },
      "texture": {
        "uri": "...",
        "type": "image/png"
        // Data URIの場合、embeddedフラグは不要（自明）
      }
    }
  }
}
```

リソースIDのスコープ:

- リソースIDは**カテゴリ内で一意**である必要があります
- 異なるカテゴリ間では同じIDを使用可能
- 参照時はカテゴリとIDの組み合わせで特定されます

注意: Data URI形式（`data:`で始まる）の場合、`embedded`フラグは不要です。URIスキームから自動的に判定されます。

リソースタイプの拡張

新しいリソースタイプを追加する場合は、`x-`プレフィックスを使用：

```json
{
  "resources": {
    // 標準リソースタイプ
    "images": {
      "texture01": {
        "uri": "./textures/detail.png",
        "type": "image/png"
      }
    },
    // カスタムリソースタイプ
    "x-particles": {
      "sparkle": {
        "uri": "./effects/sparkle.json",
        "type": "application/json",
        "schema": "https://example.com/particle-schema.json"
      }
    },
    "x-behaviors": {
      "enemyAI": {
        "uri": "./scripts/enemy_ai.lua",
        "type": "text/x-lua"
      }
    }
  }
}
```

標準リソースタイプ

タイプ	用途	MIMEタイプ例
images	テクスチャ、参照画像	image/png, image/jpeg
sounds	効果音、BGM	audio/ogg, audio/mp3
scripts	シェーダー、スクリプト	text/x-glsl, application/javascript
fonts	カスタムフォント	font/ttf, font/woff2
shaders	描画エフェクト	text/x-glsl, text/x-hlsl
data	汎用データ	application/json, application/xml

URI形式

形式	例	説明
相対パス	`./images/tex.png`	MRPAFファイルからの相対
絶対URL	`https://example.com/tex.png`	外部リソース
Data URI	`...`	埋め込みデータ

圧縮とエンコーディング

ピクセルデータの構造

1. 配列形式（array）

```
json
```

```
{
  "pixels": {
    "encoding": "array",
    "data": [
      [0, 1, 2, 3],  // 行0
      [4, 5, 6, 7],  // 行1
      [8, 9, 0, 1]   // 行2
    ]
  }
}
```

2. ランレングス圧縮（rle）

```
json
```

```
{
  "pixels": {
    "encoding": "rle",
    "data": "0:10,1:5,2:3,0:8|1:4,2:12,3:8"
    // 形式: "色ID:個数,..." 各行を"|"で区切り
  }
}
```

3. スパース配列（sparse）

```json
{
  "pixels": {
    "encoding": "sparse",
    "dimensions": {"width": 48, "height": 48},
    "defaultValue": 0,
    "data": [
      {"x": 10, "y": 15, "color": 2},
      {"x": 11, "y": 15, "color": 2},
      {"x": 12, "y": 15, "color": 3}
    ]
  }
}
```

4. インデックス化RLE（indexed-rle）

```json
{
  "pixels": {
    "encoding": "indexed-rle",
    "palette": [0, 2, 5, 7],  // 使用する色IDのみ
    "data": "0:10,1:5,2:3,0:8"  // パレットインデックスを使用
  }
}
```

5. デルタ圧縮（delta）

```json
{
  "pixels": {
    "encoding": "delta",
    "baseFrame": 0,  // アニメーションフレームのインデックス
    "data": [
      {"x": 10, "y": 5, "from": 1, "to": 2},
      {"x": 11, "y": 5, "from": 1, "to": 3}
    ]
  }
}
```

6. 領域ベース圧縮（regions）

```json
{
  "pixels": {
    "encoding": "regions",
    "dimensions": {"width": 32, "height": 32},
    "data": [
      {
        "x": 0, "y": 0,
        "width": 16, "height": 16,
        "encoding": "rle",
        "data": "0:256"
      },
      {
        "x": 16, "y": 0,
        "width": 16, "height": 16,
        "encoding": "array",
        "data": [[1,2], [3,4]]
      }
    ]
  }
}
```

圧縮プロファイル

圧縮設定の優先順位（高い順）：

1. **レイヤーの明示的な`encoding`指定**
2. **レイヤーレベルの`compressionProfile`**
3. **ルートレベルの`compressionProfile`**
4. **システムデフォルト（"array"）**

```json
{
  // ルートレベル（全レイヤーのデフォルト）
  "compressionProfile": {
    "name": "balanced",
    "settings": {
      "targetSize": "optimal",
      "quality": 0.95,
      "speed": "normal",
      "autoSelect": true  // encodingが"auto"の場合に適用
    }
  },
  "layers": [
    {
      "pixels": {
        // ケース1: 明示的指定（最優先）
        "encoding": "sparse"
      }
    },
    {
      "pixels": {
        // ケース2: 自動選択（compressionProfileを使用）
        "encoding": "auto"
      }
    },
    {
      "pixels": {
        // ケース3: レイヤー固有のプロファイル
        "encoding": "auto",
        "compressionProfile": {
          "name": "highDetail",
          "settings": {
            "quality": 1.0,
            "preferredEncodings": ["sparse", "indexed-rle"]
          }
        }
      }
    }
  ]
}
```

autoSelectの動作:

- `encoding: "auto"`が指定された場合のみ有効
- レイヤーの特性を分析して最適なエンコーディングを選択

- `preferredEncodings`で使用可能な圧縮方式を制限（優先順位順のリスト）
  - 例: `["sparse", "indexed-rle"]` → sparseを優先的に試し、適さない場合はindexed-rleを使用
  - リストにない圧縮方式は使用されません

ピクセルデータ圧縮の自動選択

```json
{
  "pixels": {
    "encoding": "auto",
    "hint": "sparse",
    "data": {}
  }
}
```

バイナリフォーマット

MessagePack版

MRPAFはMessagePackバイナリ形式もサポート:

```
[Header]
0x4D 0x52 0x50 0x41 0x46  // "MRPAF"
0x02 0x00 0x00           // Version 2.0.0
[MessagePack Data]
```

利点:

- JSONの30-70%のサイズ
- 高速な読み書き
- 型情報の保持

既存ツールとの連携

Aseprite エクスポータ

lua

```lua
-- Aseprite script for MRPAF export
function exportMRPAF(sprite)
  local mrpaf = {
    format = "MRPAF",
    version = "2.0.0",
    metadata = {
      title = sprite.filename,
      tool = {
        name = "Aseprite",
        version = app.version,
        exporter = "MRPAF-Aseprite v1.0"
      }
    }
    -- レイヤー変換ロジック
  }
  return mrpaf
end
```

変換マッピング

Aseprite	MRPAF	備考
Layer	layers[]	グループ構造を保持
Frame	animations.frames[]	タグ情報も移行
Slice	resources.slices	9スライス対応

実装ガイド

バージョン互換性

互換性マトリックス

パーサーバージョン	読み込み可能なフォーマット
2.0.0	2.0.0
2.0.1	2.0.0, 2.0.1
2.1.0	2.0.0, 2.0.1, 2.1.0

v2.0.0からv2.0.1への移行

v2.0.0ファイルを読み込む際の自動変換：

1. `layers[].placement.width/height`は無視
2. `layers[].pixels.data.values`（sparse）→`layers[].pixels.data`に平坦化
3. `animations[].frames[].index`は無視

4. `events[].resource`（ファイル名）→ リソースIDに変換

バージョンチェックの実装

```javascript
class MRPAFParser {
  parseVersion(versionString) {
    const [major, minor, patch] = versionString.split('.').map(Number);
    return { major, minor, patch };
  }

  isCompatible(documentVersion) {
    const parserVersion = this.parseVersion('2.0.1');
    const docVersion = this.parseVersion(documentVersion);

    // メジャーバージョンが異なる場合は非互換
    if (docVersion.major !== parserVersion.major) {
      return false;
    }

    // マイナーバージョンがパーサーより新しい場合は非互換
    if (docVersion.minor > parserVersion.minor) {
      return false;
    }

    // パッチバージョンは互換性に影響しない
    return true;
  }
}
```

パーサー実装の基本フロー

```javascript
class MRPAFParser {
  async parse(file) {
    // 1. フォーマット検証
    const data = JSON.parse(file);
    if (data.format !== 'MRPAF') {
      throw new Error('Invalid format');
    }

    // 2. バージョン互換性チェック
    if (!this.isCompatible(data.version)) {
      throw new Error(`Unsupported version: ${data.version}`);
    }

    // 3. スキーマ検証（オプション）
    if (data.$schema) {
      await this.validateSchema(data);
    }

    // 4. データ構築
    return this.buildDocument(data);
  }

  isCompatible(versionString) {
    const docVersion = this.parseVersion(versionString);
    const parserVersion = this.parseVersion('2.0.1');

    // メジャーバージョンチェック
    if (docVersion.major !== parserVersion.major) return false;

    // マイナーバージョンチェック
    if (docVersion.minor > parserVersion.minor) return false;

    return true;
  }
}
```

レンダラー実装のポイント

マルチ解像度の座標変換

```javascript
function transformCoordinate(layerCoord, layer) {
  const baseCoord = {
    x: layer.placement.x + (layerCoord.x / layer.resolution.scale),
    y: layer.placement.y + (layerCoord.y / layer.resolution.scale)
  };
  return baseCoord;
}
```

効率的なレイヤー合成

```javascript
class LayerCompositor {
  composite(layers, canvas) {
    // レイヤーをZオーダーでソート
    const sortedLayers = layers.sort((a, b) => a.id - b.id);

    for (const layer of sortedLayers) {
      if (!layer.visible) continue;

      // 解像度に応じた最適化レンダリング
      if (layer.resolution.scale >= 4) {
        this.renderHighResolution(layer, canvas);
      } else {
        this.renderStandard(layer, canvas);
      }
    }
  }
}
```

パフォーマンス最適化

1. **遅延デコード**: 表示に必要なレイヤーのみデコード
2. **キャッシング**: デコード済みピクセルデータをキャッシュ
3. **LOD対応**: ズームレベルに応じて解像度を切り替え

ツールとエコシステム

サンプルファイル

最小構成

json

```json
{
  "$schema": "https://mrpaf.org/schemas/v2.0.1/mrpaf.schema.json",
  "format": "MRPAF",
  "version": "2.0.1",
  "metadata": {
    "title": "Simple Sprite"
  },
  "canvas": {
    "baseWidth": 16,
    "baseHeight": 16
  },
  "palette": [
    {"id": 0, "hex": "#00000000"}
  ],
  "layers": [
    {
      "id": 0,
      "name": "Base",
      "resolution": {
        "pixelArraySize": {"width": 16, "height": 16},
        "scale": 1.0
      },
      "placement": {
        "x": 0,
        "y": 0
      },
      "pixels": {
        "encoding": "array",
        "data": []
      }
    }
  ]
}
```

マルチ解像度＋アニメーション完全例

```
json
```

```
{
  "$schema": "https://mrpaf.org/schemas/v2.0.0/mrpaf.schema.json",
  "format": "MRPAF",
  "version": "2.0.0",
  "metadata": {
    "title": "アニメーションキャラクター",
    "author": "PixelArtist",
    "created": "2024-01-15T10:00:00Z",
    "tags": ["character", "rpg", "animated"]
  },
  "canvas": {
    "baseWidth": 32,
    "baseHeight": 32,
    "pixelUnit": 1.0,
    "backgroundColor": "#00000000"
  },
  "coordinateSystem": {
    "origin": "top-left",
    "xAxis": "right",
    "yAxis": "down",
    "baseUnit": 1.0,
    "allowFloatingPoint": true,
    "subPixelPrecision": 4
  },
  "compressionProfile": {
    "name": "balanced",
    "settings": {
      "autoSelect": true,
      "quality": 0.95
    }
  },
  "palette": [
    {"id": 0, "name": "透明", "hex": "#00000000"},
    {"id": 1, "name": "肌", "hex": "#FDBCB4"},
    {"id": 2, "name": "髪", "hex": "#8B4513"},
    {"id": 3, "name": "服", "hex": "#4169E1"},
    {"id": 4, "name": "目", "hex": "#000000"},
    {"id": 5, "name": "白目", "hex": "#FFFFFF"}
  ],
  "layers": [
    {
      "id": 0,
      "name": "ベースボディ",
      "resolution": {
        "pixelArraySize": {"width": 32, "height": 32},
        "scale": 1.0,
```

```
      "effectiveSize": {"width": 32, "height": 32}
    },
    "placement": {
      "x": 0, "y": 0,
      "width": 32, "height": 32,
      "unit": "base"
    },
    "pixels": {
      "encoding": "rle",
      "data": "0:32|0:8,1:16,0:8|..."
    }
  },
  {
    "id": 1,
    "name": "顔ディテール",
    "resolution": {
      "pixelArraySize": {"width": 32, "height": 16},
      "scale": 4.0,
      "effectiveSize": {"width": 8, "height": 4}
    },
    "placement": {
      "x": 12,
      "y": 8,
      "width": 8,
      "height": 4,
      "unit": "base",
      "allowSubPixel": true
    },
    "pixels": {
      "encoding": "sparse",
      "dimensions": {"width": 32, "height": 16},
      "defaultValue": 0,
      "data": [
        {"x": 8, "y": 4, "color": 4},
        {"x": 9, "y": 4, "color": 5},
        {"x": 24, "y": 4, "color": 4},
        {"x": 25, "y": 4, "color": 5}
      ]
    }
  }
],
"animations": {
  "idle": {
    "fps": 2,
    "loops": true,
    "frames": [
      {
```

```
        "duration": 500,
        "layers": [0, 1]
      },
      {
        "duration": 100,
        "layers": [0],
        "overrides": {
          "1": {"visible": false}
        }
      }
    ],
    "tags": ["character", "idle"]
  },
  "walk": {
    "fps": 8,
    "loops": true,
    "frames": [
      {
        "duration": 125,
        "layers": [0, 1],
        "tweens": {
          "0": {
            "placement": {
              "y": {"from": 0, "to": -0.5, "easing": "ease-out"}
            }
          }
        },
        "events": [
          {"type": "sound", "resourceId": "footstep", "time": 0}
        ]
      },
      {
        "duration": 125,
        "layers": [0, 1],
        "tweens": {
          "0": {
            "placement": {
              "y": {"from": -0.5, "to": 0, "easing": "ease-in"}
            }
          }
        }
      }
    ],
    "tags": ["character", "movement"]
  }
},
"animationController": {
```

```
    "defaultAnimation": "idle",
    "transitions": [
      {
        "from": "idle",
        "to": "walk",
        "condition": "moving",
        "duration": 100
      }
    ]
  },
  "resources": {
    "sounds": {
      "footstep": {
        "uri": "./sounds/step.ogg",
        "type": "audio/ogg"
      }
    }
  }
}
```

ツールとエコシステム

公式実装

mrpaf-js (JavaScript/TypeScript)

```bash
npm install mrpaf-js
```

```javascript
import { MRPAFDocument } from 'mrpaf-js';

const doc = await MRPAFDocument.load('character.mrpaf');
const canvas = doc.render();
```

mrpaf-rust (Rust/WebAssembly)

```toml
[dependencies]
mrpaf = "2.0"
```

エディタプラグイン

Aseprite Extension

```lua
-- MRPAFエクスポート機能を追加
local mrpaf = require("mrpaf-aseprite")
mrpaf.export(activeSprite, {
  multiResolution = true,
  compression = "auto"
})
```

Photoshop プラグイン

- レイヤーグループを解像度レベルとして認識
- スマートオブジェクトからの自動変換

ゲームエンジン統合

Unity Package

```csharp
using MRPAF.Unity;

[CreateAssetMenu]
public class MRPAFImporter : ScriptedImporter {
    public override void OnImportAsset(AssetImportContext ctx) {
        var doc = MRPAFDocument.Load(ctx.assetPath);
        var sprites = doc.GenerateSprites();
        // Unityアセットとして登録
    }
}
```

Godot アドオン

```gdscript
extends EditorImportPlugin

func import(source_file, save_path, options, r_platform_variants, r_gen_files):
    var mrpaf = MRPAF.new()
    mrpaf.load(source_file)
    return mrpaf.to_godot_resources()
```

オンラインツール

- **MRPAFビューア**: https://viewer.mrpaf.org
- **コンバーター**: https://convert.mrpaf.org

- **バリデーター**: https://validate.mrpaf.org

ベストプラクティス

解像度設計のガイドライン

1. **キャラクターデザイン**
   - ベース: 16x16 または 32x32
   - 顔の詳細: 4倍解像度
   - アクセサリー: 2倍解像度
2. **背景アート**
   - ベース: 画面解像度の1/4
   - 前景オブジェクト: 1倍解像度
   - インタラクティブ要素: 2倍解像度
3. **UI要素**
   - アイコン: 1倍解像度
   - テキスト領域: 2-4倍解像度
   - ボタン: 9スライス + 2倍解像度

圧縮戦略

```json
{
  "compressionProfile": {
    "rules": [
      {
        "condition": "layer.scale >= 4 && layer.coverage < 0.3",
        "encoding": "sparse"
      },
      {
        "condition": "layer.colorCount <= 4",
        "encoding": "indexed-rle"
      },
      {
        "condition": "layer.type === 'background'",
        "encoding": "base64"
      }
    ]
  }
}
```

アニメーション設計

1. **キーフレームの配置**
   - 主要ポーズ: 通常解像度
   - 表情変化: 高解像度レイヤー
   - エフェクト: 別レイヤーで管理
2. **パフォーマンス考慮**
   - 60fps: 重要なアクションのみ
   - 30fps: 通常のキャラクター動作
   - 15fps: 背景アニメーション

トラブルシューティング

よくある問題と解決法

Q: 高解像度レイヤーがぼやける

A: `blending.interpolation` を `"nearest"` に設定

Q: ファイルサイズが大きい

A: 圧縮プロファイルで `autoSelect: true` を有効化

Q: アニメーションがカクカクする

A: サブピクセル精度を有効化し、イージング関数を適用

今後のロードマップ

v2.1 (2024 Q2)

- 3Dレイヤーサポート
- リアルタイムコラボレーション機能
- AI支援圧縮

v3.0 (2024 Q4)

- ベクターレイヤーとの統合
- プロシージャル生成サポート
- ブロックチェーン統合

コミュニティとサポート

公式チャンネル

- GitHub: https://github.com/mrpaf/specification

- Discord: https://discord.gg/mrpaf
- Forum: https://forum.mrpaf.org

貢献方法

1. 仕様への提案: GitHub Issuesで議論
2. 実装の共有: mrpaf-contribリポジトリ
3. ドキュメント改善: Pull Request歓迎

ライセンスと貢献

仕様ライセンス

参照実装ライセンス

- mrpaf-js: MIT License
- mrpaf-rust: MIT/Apache-2.0

貢献者

- 仕様設計: MRPAF Working Group
- 初期実装: [貢献者リスト]

付録

用語集

- **効果的解像度**: ベース座標系でのレイヤーの実際のサイズ
- **サブピクセル**: 1ピクセル未満の精度での座標指定
- **スパース配列**: 非ゼロ要素のみを保存する配列形式

参考文献

- [1] "Efficient Multi-Resolution Image Formats" - Computer Graphics Forum
- [2] "Sub-pixel Precision in 2D Animation" - SIGGRAPH Papers
- [3] "Modern Pixel Art Techniques" - Game Developer Magazine