Origami Engine

A GameMaker Studio 1.4-inspired game engine for TypeScript and the web

Build 2D platformers and games with familiar GameObject-based APIs, running natively in modern browsers with full TypeScript support.

Status: ✅ MVP Complete - Fully functional with working platformer example!

🎮 Features

- **Familiar GMS-style API** - GameObject classes with `create()`, `step()`, `draw()` events
- **TypeScript First** - Full type safety with strict mode support
- **Browser Native** - Runs in any modern browser using Canvas 2D
- **Easy to Learn** - Similar to GameMaker Studio 1.4.9999
- **Complete Tooling** - CLI for project creation, dev server, and builds
- **Debug Tools** - Built-in FPS counter, collision visualization (press F3)
- **Working Example** - Full platformer game included with source

---

🚀 Quick Start

For This Repository (Developers/Contributors)

This is a monorepo containing the runtime engine, CLI tool, and example game.

```
# Clone or navigate to repository
cd Origami-Engine

# Install dependencies (requires pnpm)
pnpm install

# Build all packages
pnpm build

# Try the example platformer
cd platformer
pnpm install
pnpm start
```

Open http://localhost:3000 in your browser and play!

Controls: WASD to move, Space to jump, F3 for debug mode

For End Users (After Publishing to npm)

Once published, create games with the CLI:

```
# Install CLI globally
npm install -g origami-cli

# Create a new game
ori create my-platformer

# Navigate and start
cd my-platformer
npm install
npm start
```

Open http://localhost:3000 and start developing!

📚 Documentation

For Users

- **Getting Started Guide** - Installation, first steps, tutorials
- **Runtime API Reference** - Complete function reference (GMS style)
- **CLI Reference** - All commands and configuration

For Contributors

- **Project Specification** - Design decisions and architecture
- **Current Status** - What's implemented and what's next
- **Package READMEs** - Runtime and CLI package documentation

PDF Documentation

All documentation is also available as PDF in **docs/pdf/**

💡 Core Concepts

GameObject Class

All game objects extend `GameObject`:

```
import { GameObject } from 'origami-runtime';

export class obj_player extends GameObject {
  create(): void {
    this.sprite_index = 'spr_player';
    this.x = 100;
    this.y = 100;
  }

  step(): void {
    // Movement
    if (keyboard_check(vk_right)) this.x += 4;
    if (keyboard_check(vk_left)) this.x -= 4;

    // Jump
    if (keyboard_check_pressed(vk_space)) {
      this.vspeed = -10;
    }

    // Gravity
    this.vspeed += 0.5;

    // Collision
    if (place_meeting.call(this, this.x, this.y + this.vspeed, 'obj_wall')) {
      this.vspeed = 0;
    }
  }

  draw(): void {
    draw_self.call(this);
  }
}
```

Built-in Properties

Every GameObject has GMS-style properties:

```
// Position & motion
x, y, xprevious, yprevious, xstart, ystart
speed, direction, hspeed, vspeed

// Sprite & animation
sprite_index, image_index, image_speed
image_alpha, image_angle, image_xscale, image_yscale

// Rendering & behavior
visible, depth, order, persistent
```

Global Functions

Familiar GameMaker functions available globally:

```
// Instance management
await instance_create(x, y, 'obj_bullet');
instance_destroy.call(this);
instance_exists('obj_player');

// Collision
place_meeting.call(this, x, y, 'obj_wall');
const coin = instance_place.call(this, x, y, 'obj_coin');

// Input
keyboard_check(vk_space);
keyboard_check_pressed(vk_w);
mouse_check_button(mb_left);

// Drawing (in draw event)
draw_sprite('spr_player', 0, x, y);
draw_text(10, 10, 'Score: 100');
draw_set_color('#FF0000');
draw_rectangle(x1, y1, x2, y2, false);

// Math & motion
point_direction(x1, y1, x2, y2);
point_distance(x1, y1, x2, y2);
lengthdir_x(speed, direction);
lengthdir_y(speed, direction);

// Room
await room_goto('room_level2');

// Utility
random(100);
irandom(10);
random_range(min, max);
```

🗂️ Project Structure

Created Game Projects

```
my-game/
├── objects/          # Game object TypeScript classes
│   ├── obj_player.ts
│   ├── obj_wall.ts
│   └── obj_enemy.ts
├── sprites/          # Sprite assets
│   └── spr_player/
│       ├── metadata.json
│       └── frame_0.png
├── rooms/            # Room definitions (JSON)
│   └── room_level1.json
├── src/              # Entry point
│   └── main.ts
├── game.json         # Game configuration
├── index.html        # HTML entry
├── package.json
└── tsconfig.json
```

This Repository

```
Origami-Engine/
├── packages/
│   ├── runtime/      # Core game engine (npm: origami-runtime)
│   └── cli/          # CLI tool (npm: origami-cli)
├── platformer/       # Complete working example game
├── docs/             # Documentation
│   ├── guides/       # Getting started, tutorials
│   ├── reference/    # Specification
│   ├── development/  # Status, contributing
│   └── pdf/          # PDF versions
├── scripts/          # PDF generation tools
└── README.md         # This file
```

🎨 Sprites & Assets

Sprites are organized in folders:

```
sprites/spr_player/
├── metadata.json
├── frame_0.png
├── frame_1.png
└── ...
```

metadata.json:

```
{
  "origin": { "x": 16, "y": 16 },
  "fps": 10
}
```

Supported formats: PNG, JPG, WebP

🏠 Rooms

Rooms are JSON files defining game levels:

```
{
  "name": "room_level1",
  "width": 640,
  "height": 480,
  "speed": 60,
  "backgroundColor": "#87CEEB",
  "instances": [
    { "object": "obj_player", "x": 100, "y": 100 },
    { "object": "obj_wall", "x": 0, "y": 400 }
  ],
  "views": [{
    "enabled": true,
    "object": "obj_player",
    "hborder": 200,
    "vborder": 150
  }]
}
```

🔧 Development

CLI Commands

```
ori create <name>    # Create new project from template
ori dev              # Start development server
ori build            # Build for production
ori --help           # Show help
```

Working on the Engine

```
# Build all packages
pnpm build

# Run example game
cd platformer
pnpm start

# Clean build artifacts
pnpm clean
```

Debug Mode

Press **F3** or **~** in-game to toggle:

- FPS counter
- Instance count
- Collision box visualization (color-coded)
- View/camera position

📦 Publishing to npm

Packages are ready for npm publishing:

```
# Publish runtime
cd packages/runtime
npm publish

# Publish CLI
cd packages/cli
npm publish
```

✅ Implementation Status

Complete and Functional:

- ✅ Core game engine with 60 FPS fixed timestep
- ✅ GameObject system with all GMS events
- ✅ Sprite system with animation
- ✅ Keyboard & mouse input
- ✅ AABB collision detection
- ✅ Room system with JSON definitions
- ✅ View/camera with deadzone following
- ✅ Canvas 2D rendering with depth sorting
- ✅ Drawing API (sprites, text, shapes, colors)
- ✅ Debug overlay and tools
- ✅ Save/load with localStorage
- ✅ Complete platformer example
- ✅ Full documentation

Ready For:

- 🎯 Publishing to npm
- 🎯 Creating games
- 🎯 Community testing and feedback

See **docs/development/STATUS.md** for detailed status.

🤝 Contributing

This is an MVP. Contributions, feedback, and bug reports are welcome!

1. Fork the repository
2. Create a feature branch
3. Make your changes
4. Submit a pull request

See **docs/reference/SPECIFICATION.md** for architecture details.

🗺️ Roadmap

Future enhancements being considered:

- Sound/audio system
- Particle effects
- Tilemap support and collision
- More collision shapes (circles, precise pixel)
- Visual room editor
- Sprite/animation editor
- Pathfinding system
- Network/multiplayer support

📝 License

MIT - See LICENSE for details

🔗 Links

- **Runtime Package** - Core engine documentation
- **CLI Package** - CLI tool documentation
- **Example Game** - Full platformer source code
- **Documentation** - Complete docs in markdown and PDF

Built with ❤️ for game developers who love GameMaker Studio 1.4