Introduction to **Game Development:**

Legend of Royal Institution

The Royal Institution
Science Lives Here

What makes up a video game

Technical aspects of developing games

Practice

Further reading and recommendations

What makes up a video game

Formulae of a game

Q: What makes game a game?

Game = ??? + ??? + ???

**Images courtesy of Sega Corporation, Xbox Game Studios, Electronic Arts Inc. (top to bottom)*

Gameplay

Describes the experience of play (what players **do**) in the game and defines the connection between a game and a player.

> *“The structures of player interaction with the game system and with other players in the game.”*

Mario Wonders:

- 2D platformer in an imaginary setting, where the character called Mario, controlled by a player, fights monsters and tries to save Princess Peach.

*Image courtesy of Nintendo Co. Ltd.

QQ:
What's your favourite game?

Mechanics

Images courtesy of Mojang Studios, DotGears Company Limited, Nintendo Co. Ltd. (top to bottom)

The rules, actions, and interactions defined in a game.

Minecraft:

- Player can destroy one world block by clicking on it.
- Monsters spawn at night.

Flappy Bird:

- Tubes collide.
- Game scrolls from one side.

The Legend of Zelda:

- Character moves to another dungeon by stepping over the edge of the screen.
- Character can hit monsters.
- Character takes limited amount of damage.

Analogy: *Mechanics are like grammar rules for a language.*

QQ:
What are the game mechanics your favourite game implements?

End goal

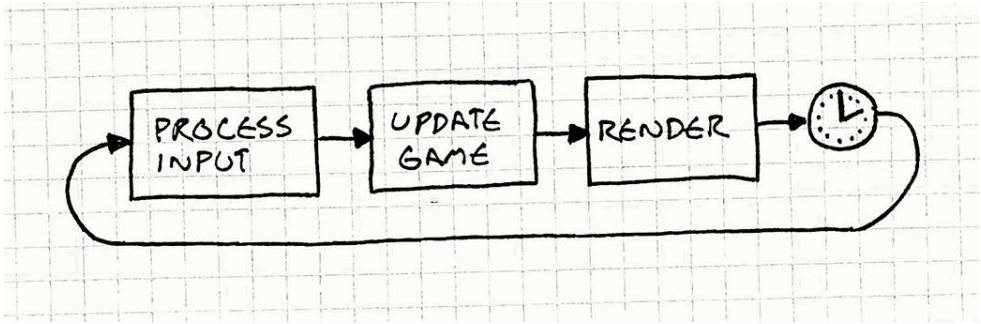
QQ: Why does a goal matter?

Crazy Taxi:
pick up and deliver as many passengers as possible

*Image courtesy of Sega Corporation

Technical aspects of developing games

Game Loop



At the heart of every game is an infinite loop, often called the **game loop**. Each cycle of this loop is known as a **frame**.

During each frame, the game engine performs three key steps:

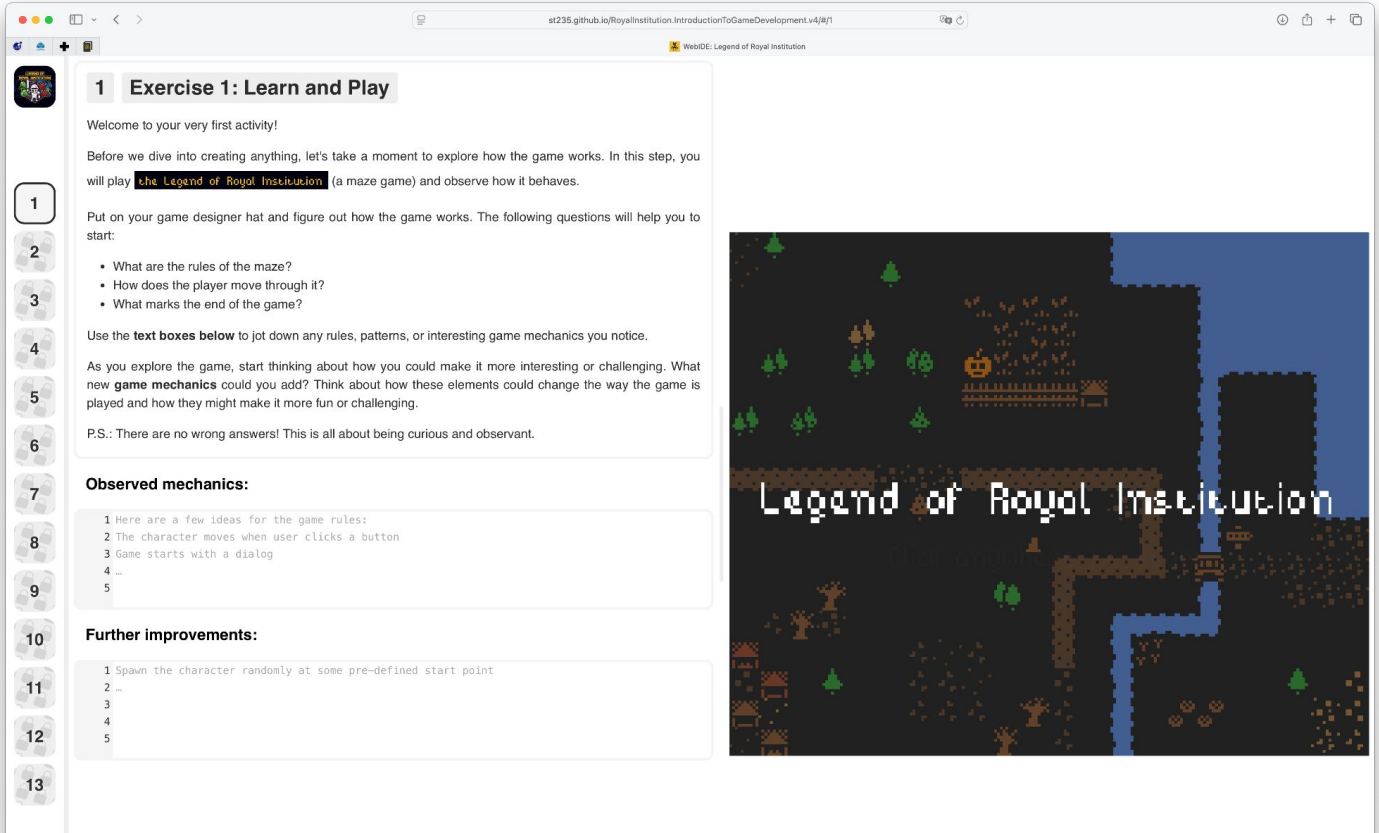
1. **Processes input/events** – like keyboard, mouse, or touch input.
2. **Updates the game state** – such as character positions, scores, and collisions.
3. **Renders the scene** – draws the current state to the screen.

**Image courtesy of "Game Programming Patterns" by Robert Nystrom*

Practice

Running activities

🔗 tinyurl.com/yzfvbyds




Learn &
Play

Data-Driven Game Development

github.com/soulsmods/Paramdex

In a **config-driven design**, the core game engine is relatively generic, while the rules, content, and behaviors are defined in external data/config files.

Dark Souls (and the other souls-like games):
extremely config/data-driven under the hood.

*Image courtesy of FromSoftware

Dark Souls 1: Config Example

```
<?xml version="1.0" encoding="utf-8"?>
<PARAMDEF XmlVersion="1">
  <ParamType>LIGHT_BANK</ParamType>
  <DataVersion>4</DataVersion>
  <BigEndian>False</BigEndian>
  <Unicode>False</Unicode>
  <FormatVersion>104</FormatVersion>
  <Fields>
    <Field Def="s16 degRotX_0">
      <DisplayName>X角度</DisplayName>
      <Description>平行光源: 0 </Description>
      <EditFlags>None</EditFlags>
      <Minimum>-90</Minimum>
      <Maximum>90</Maximum>
      <SortID>1</SortID>
    </Field>
    <Field Def="s16 degRotY_0">
      <DisplayName>Y角度</DisplayName>
      <Description>平行光源: 0 </Description>
      <EditFlags>None</EditFlags>
      <Minimum>-180</Minimum>
      <Maximum>180</Maximum>
      <SortID>2</SortID>
    </Field>
    <Field Def="s16 colR_0 = 255">
```

Data-Driven Game Development

Minecraft's gameplay is also not just defined by code in the engine: a lot of it lives in external JSON files, configs, and resource packs.

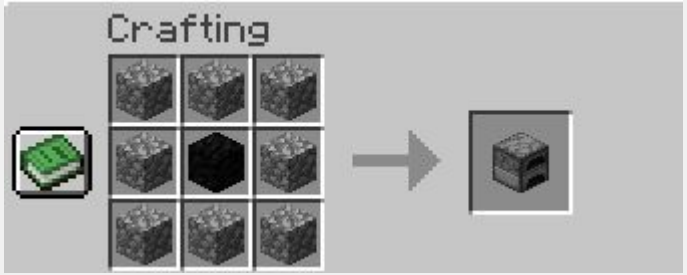
Data Packs
Introduced in **Minecraft 1.13+**, data packs define game mechanics, loot, recipes, and structures through JSON configs, such as, recipes, loot tables, or structures.

*Image courtesy of Mojang Studios

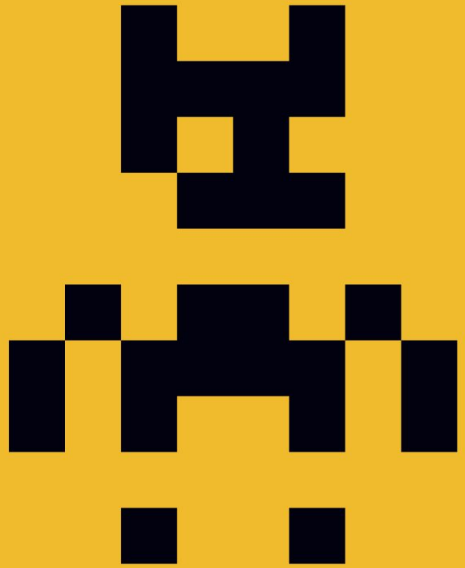
Minecraft: Config Example

```
{
  "type": "minecraft:crafting_shapeless",
  "category": "misc",
  "ingredients": [
    "minecraft:ender_pearl",
    "minecraft:blaze_powder",
    "minecraft:blaze_powder"
  ],
  "result": {
    "count": 1,
    "id": "minecraft:ender_eye"
  }
}
```

```
{
    "type": "minecraft:crafting_shaped",
    "category": "misc",
    "key": {
        "S": "#minecraft:stone_crafting_materials",
        "C": "minecraft:coal_block"
    },
    "pattern": [
        "SSS",
        "SCS",
        "SSS"
    ],
    "result": {
        "count": 1,
        "id": "minecraft:furnace"
    }
}
```



Data-Driven Game Development



The Legend of Royal Institution also relies on config files.

The configs define almost everything:

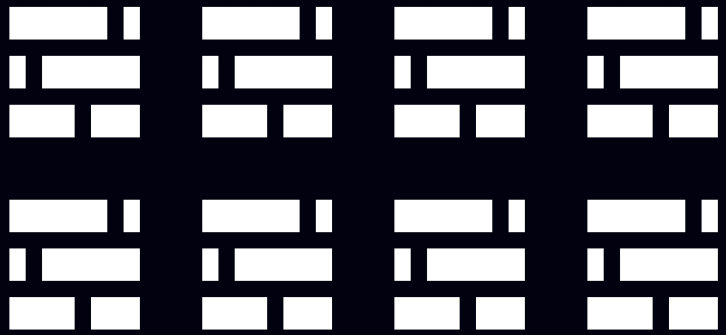
- Character appearance
- Finish (flag) position appearance
- Layout of the level

and many more...

Legend of Royal Institution: Config Example

```
{
    "gameConfig": {
        "applyCathodRayTubeEffect": false,
        "messagesOverwrites": {
            "character": 0,
            "intro": "Greetings,\nfearless students!\nEscape the maze...\nor at least debug it.",
            "introHeight": 80,
            "gameOver": "Mistakes happen...\nGive it another go",
            "victory": "Great job!\nYou've completed\nthe first step.\nMore challenges\nlie ahead!",
            "victoryHeight": 90
        },
        "configsOverwrites": {
            "walls": {
                "variations": [340, 212, 341, 339, 338, 337, 344, 342, 343, 59]
            }
        }
    },
    "initialLevelId": 1,
    "levels": [
        {
            "id": 1,
            "title": "Workshop demo",
            "levelLayout": [
                "SP .  W0 .  .  W0 .  .  .  W0 .  .  .  W0 .  W3 .  .  .  W0",
                ".  W0 W2 .  W0 W0 .  W0 .  W0 .  W0 .  .  .  .  W2 .  W0 . ",
                ".  .  .  .  .  .  .  W0 .  W0 .  W0 W0 W0 .  .  .  .  W0 . ",
                ".  W7 W7 W7 W7 .  W0 .  .  .  .  .  .  W0 W0 W0 .  W4 W0 . ",
                ".  .  .  .  W0 .  W0 W0 W0 .  W0 W0 .  .  .  .  .  .  .  . ",
                "W0 W0 W0 .  W0 .  .  .  .  .  W0 .  W0 W0 W0 W0 W7 W3 W7 W0",
                ".  .  .  .  W0 W0 W0 .  W0 .  W0 .  .  .  .  W1 .  .  .  . ",
                ".  W0 W0 .  .  .  .  .  W0 .  W0 W0 W7 .  W0 W0 W0 W0 W0 . ",
                ".  .  .  W0 W0 W0 W0 .  W0 .  .  .  .  .  W0 .  .  .  .  F0",
                ".  W0 .  .  .  .  .  .  W0 W0 W1 .  W7 .  W0 .  W0 W0 .  . ",
                ".  W0 W0 W0 W0 .  W0 W0 W0 .  .  .  W1 .  .  .  .  W2 W0 . ",
                ".  .  .  .  .  .  .  .  .  .  W1 W8 .  W0 W0 .  .  .  .  F0",
                ".  W3 W3 W3 W3 W3 W3 W3 W3 .  .  .  .  .  .  W0 W2 W0 W0 . ",
                ".  .  .  .  .  .  .  .  W0 W0 .  W0 W0 W0 .  .  .  .  .  . "
            ],
```


Time to
tune
configs



1. Level Layout

Tilemap

Tile is a small, reusable graphic (usually square or rectangular) that represents a piece of a larger map, environment, or grid in a game.

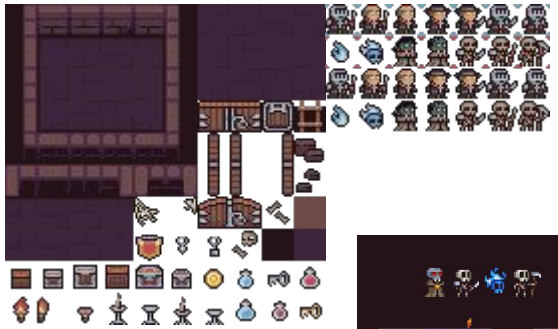
A **tilemap** is a grid-based system (usually, 2D) that arranges tiles to form game levels, environments, or maps.

There are a lot of games using tilemaps: ***Terraria, Stardew Valley, and Pokemon Ruby***.

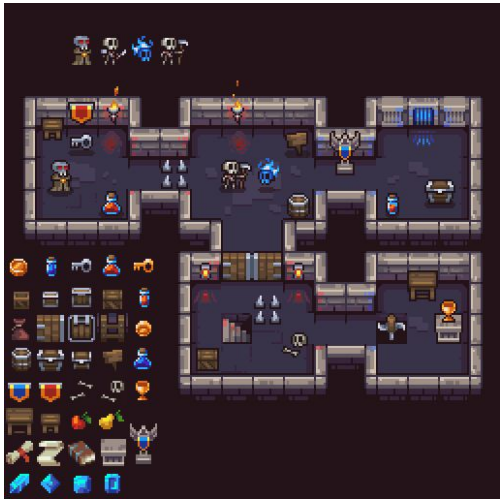
**Images courtesy of 505 Games S.p.A., ConcernedApe, Nintendo Co. Ltd. (top to bottom)*

Tiles organisation: tilesets

pixel-poem.itch.io/dungeon-assetpuck



Tileset



Level example

To keep things efficient, games use a **tileset** (a collection of all available tile graphics).

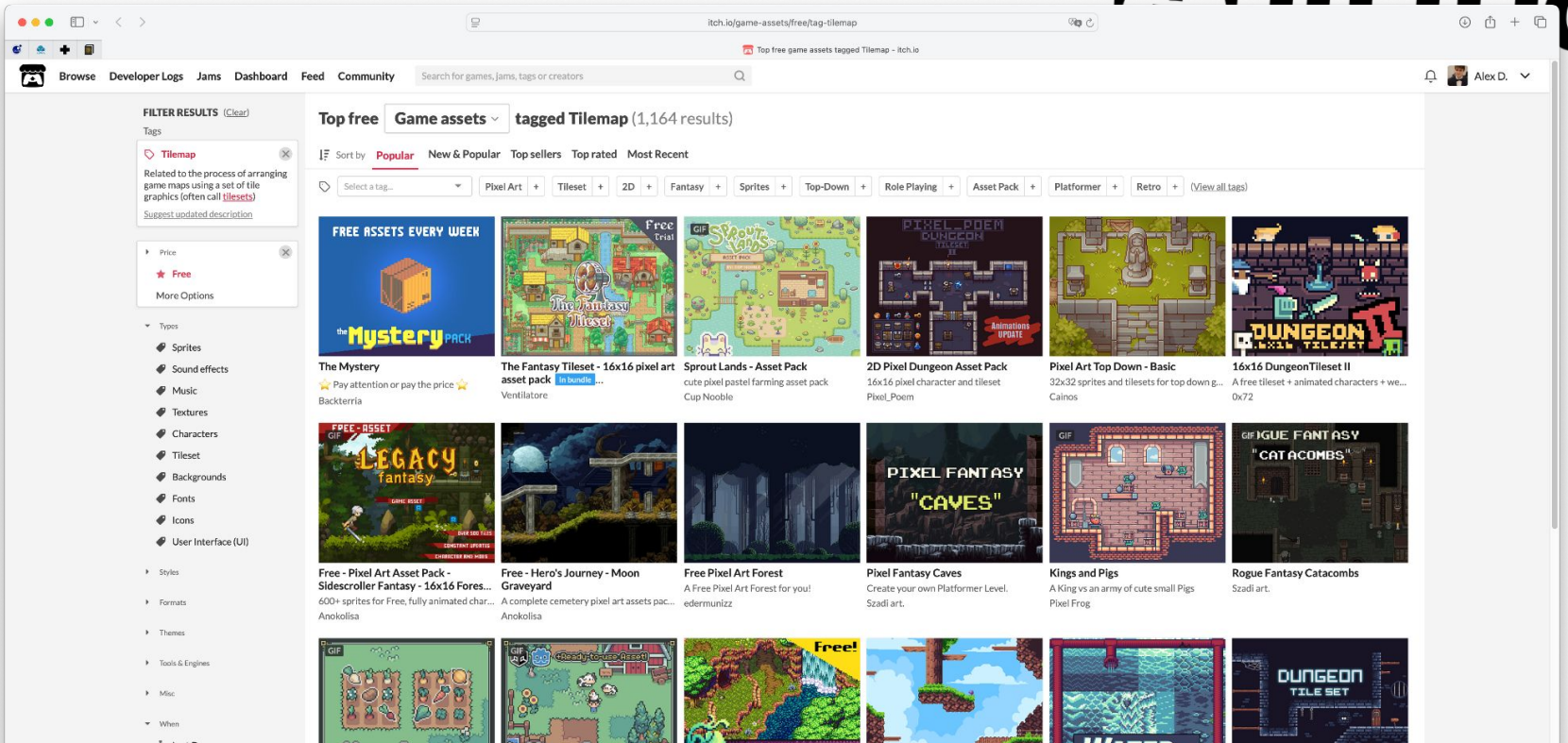
The tilemap stores only the **index or ID** of which tile appears in **each grid cell**.

This allows developers to build large environments without duplicating image data.

Often, tilemaps are arranged in layers: *a background layer for terrain, a collision layer for walls and obstacles, and a decorative layer for details.*

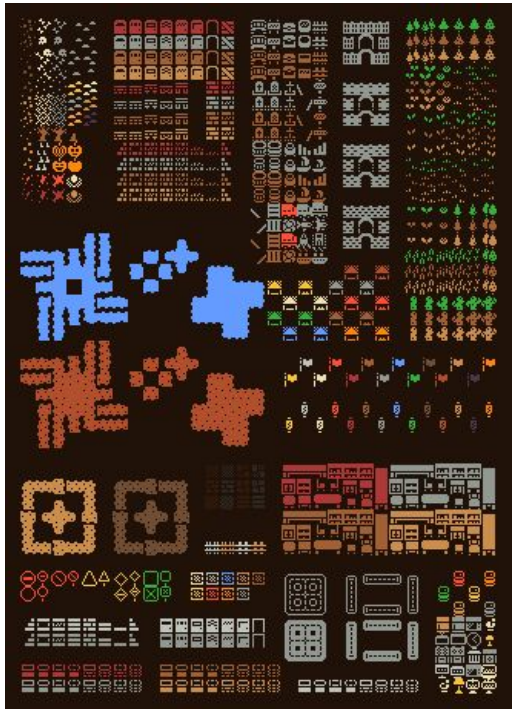
Getting assets and publishing indie projects

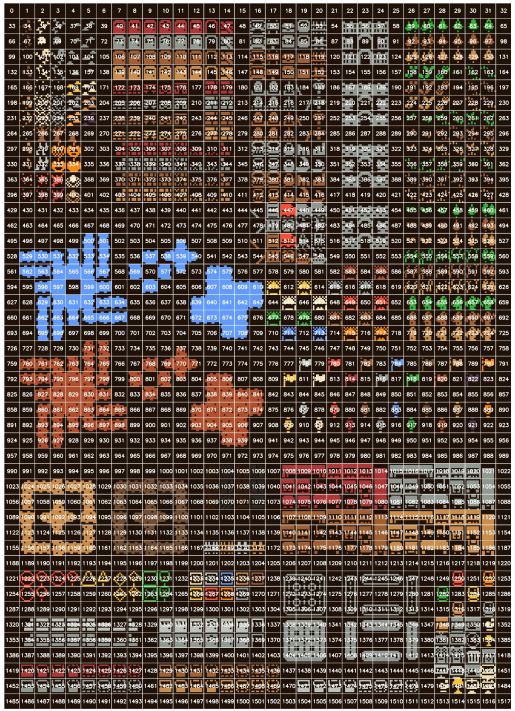
🔗 itch.io



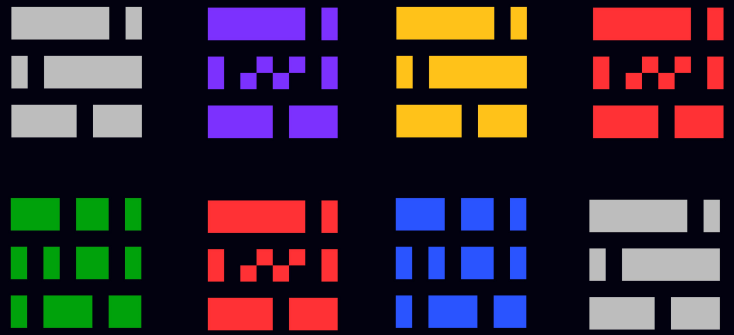
Legend of Royal Institution: tileset

v3x3d.itch.io/bountiful-bits



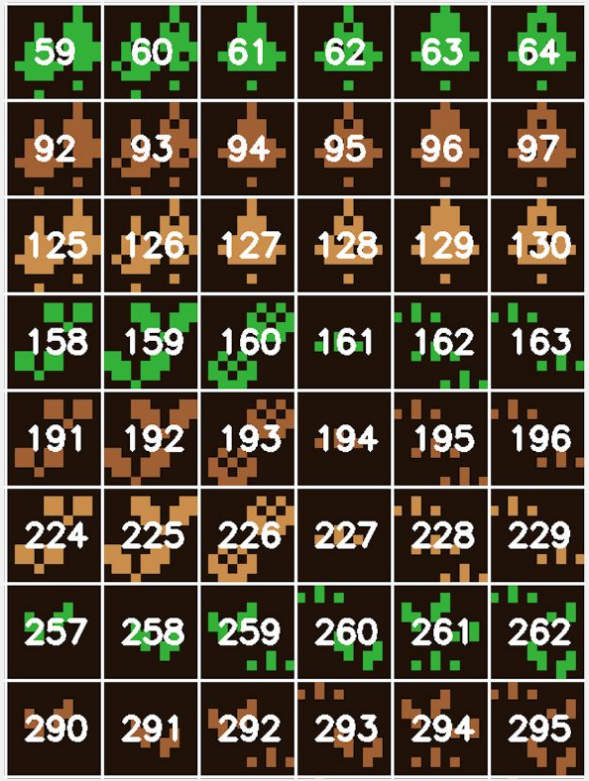


Our game also uses tilesets from itch.io with non-restrictive license.



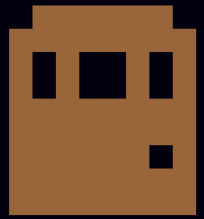
2. Walls Appearance

Walls tileset



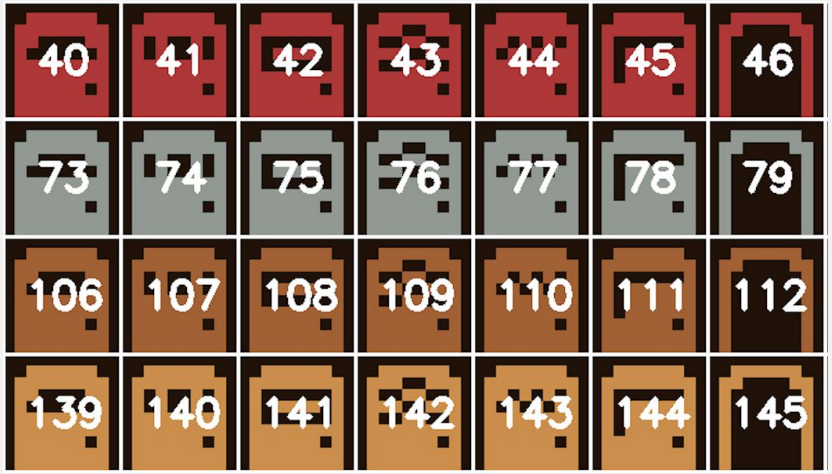
3. Coins

Coins tileset



4. Doors

Doors&Keys tilesets



5. Monsters AI

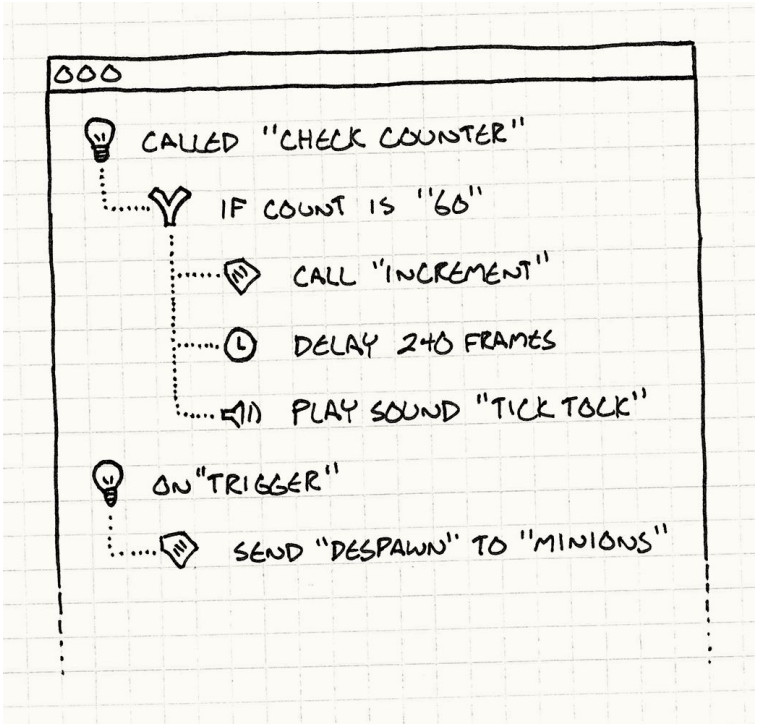
Scripting Language: Bytecode

Game can roughly be divided into a **core/engine** code and a **gameplay** code.

Engine:
- very generic
- extremely complicated
- rendering your game on a specific platform

Gameplay:
- game-specific behavior
- requires constant iteration

That's why game engines usually abstract gameplay logic into something more *lightweight*.

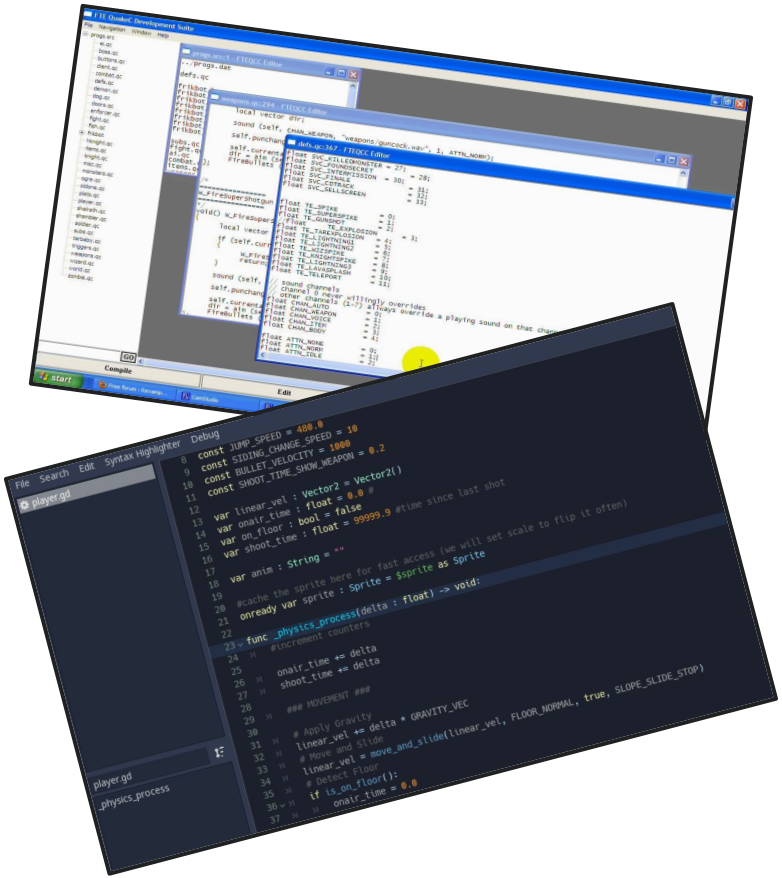


Images are courtesy of "Game Programming Patterns" by Robert Nystrom https://gameprogrammingpatterns.com/bytecode.html

Scripting Language: Custom Language

The scripting languages can be as complicated as a full-scale language within the game engine.

- **QuakeC**
- **UnrealScript**
- **GDScript**
- **Naughty Dog's DC Language**



Scripting Language: Custom Instruction Set

```
LITERAL 0    [0]            # Wizard index
LITERAL 0    [0, 0]         # Wizard index
GET_HEALTH   [0, 45]        # getHealth()
LITERAL 0    [0, 45, 0]     # Wizard index
GET_AGILITY  [0, 45, 7]     # getAgility()
LITERAL 0    [0, 45, 7, 0]  # Wizard index
GET_WISDOM   [0, 45, 7, 11] # getWisdom()
ADD          [0, 45, 18]    # Add agility and wisdom
LITERAL 2    [0, 45, 18, 2] # Divisor
DIVIDE       [0, 45, 9]     # Average agility and wisdom
ADD          [0, 54]        # Add average to current health
SET_HEALTH   []             # Set health to result
```

Alternatively, they can be defined as a **custom limited size instructions set**. For example, a spells list for a magic game.

Henry Hatsworth in the Puzzling Adventure for **Nintendo DS** used this approach.

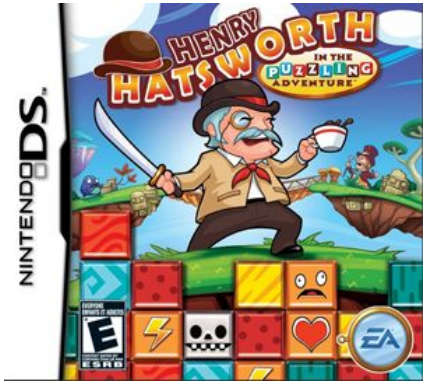


**Image is a courtesy of Electronic Arts Inc.*

Scripting Language: Custom Instruction Set

The Legend of Royal Institution uses a small markup language for monsters movement system defined by a set of rules.

A monster's movement is defined by 2 parameters:

1. **the update time (in milliseconds):** how often the monster will move to the next position.
2. **a movement list:** a set of instructions defining the specific path/trajectory of the monster.

Legend of Royal Institution: Movement System

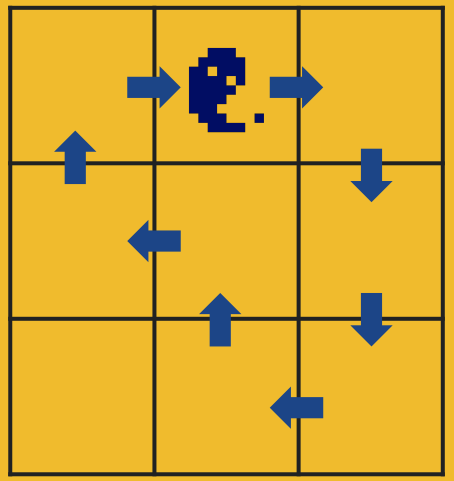
Every move command is either:

1. A direction;
2. A direction followed by **colon** and a **repeat counter**.

Allowed directions are: **left, right, up, down.**

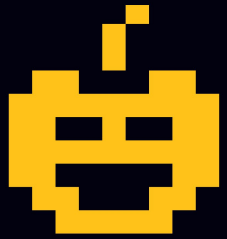
A few tips:

- Every path should **start** and **end** at **the exact same point**.
- Monsters can move **through the walls.**



Path is: right down:2 left up left up right

Creatures tileset



6. Garnitures

Easter Eggs: Hidden Locations

Easter eggs are secret features deliberately placed by developers to surprise and reward curious players.

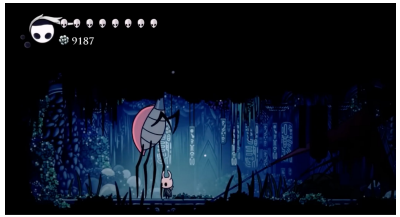
Examples:

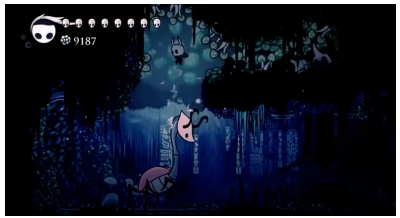
- **hidden rooms**
- references to other games
- humorous messages
- powerful items (sometimes imbalanced!)

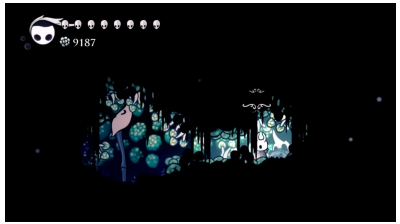
Encourage exploration and create memorable “aha!” moments.

**Image courtesy of Team Cherry*

Easter Eggs: Hidden Locations





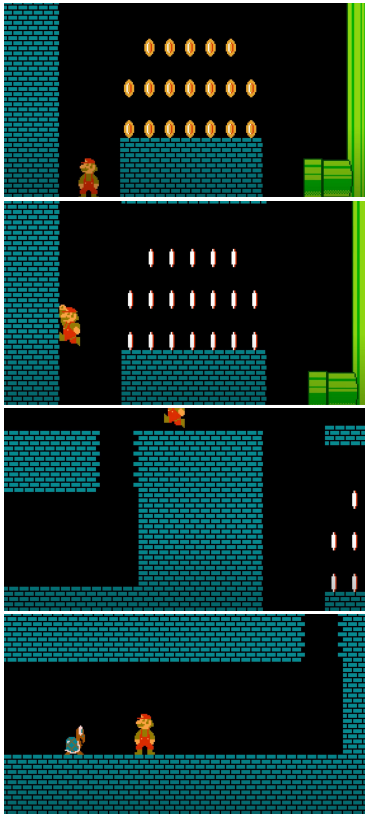


Hollow Knight

Mario Odyssey

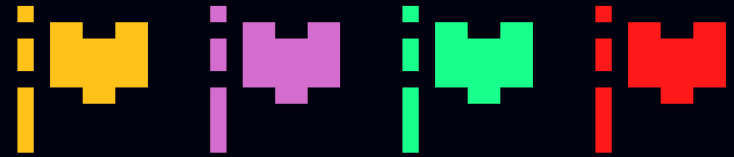
Mario 3D World

Legend Of Zelda



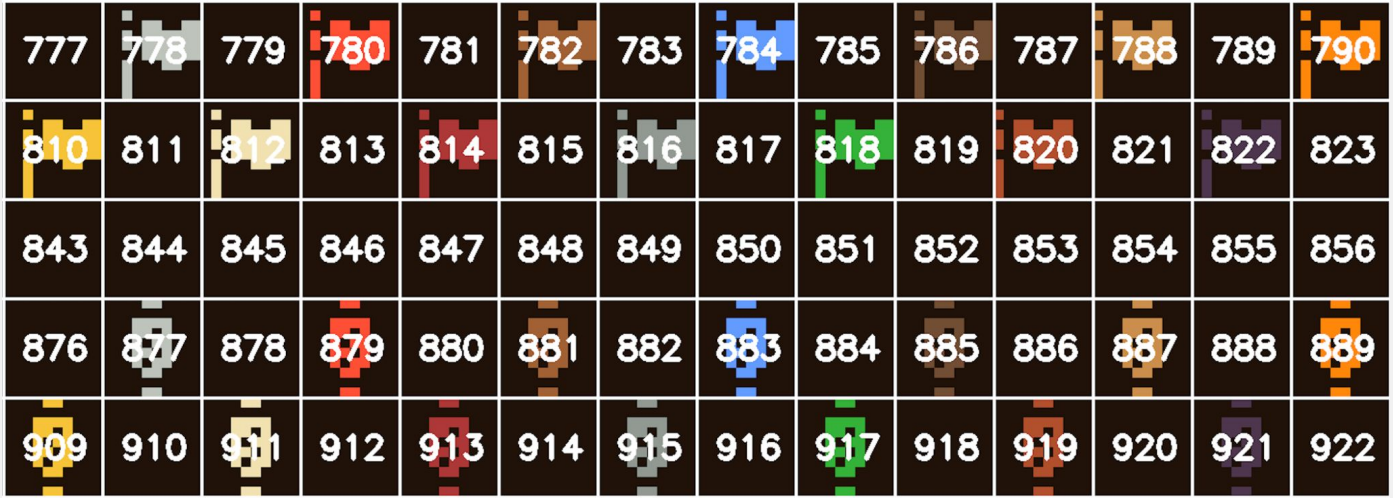
Super Mario Bros

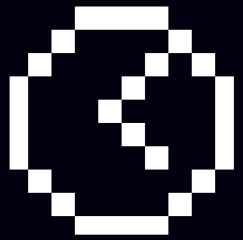
Images courtesy of Team Cherry, Nintendo Co. Ltd. (left to right)



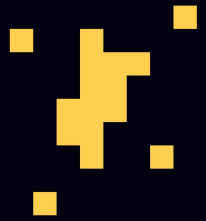
7. Flag Appearance

Flag tileset



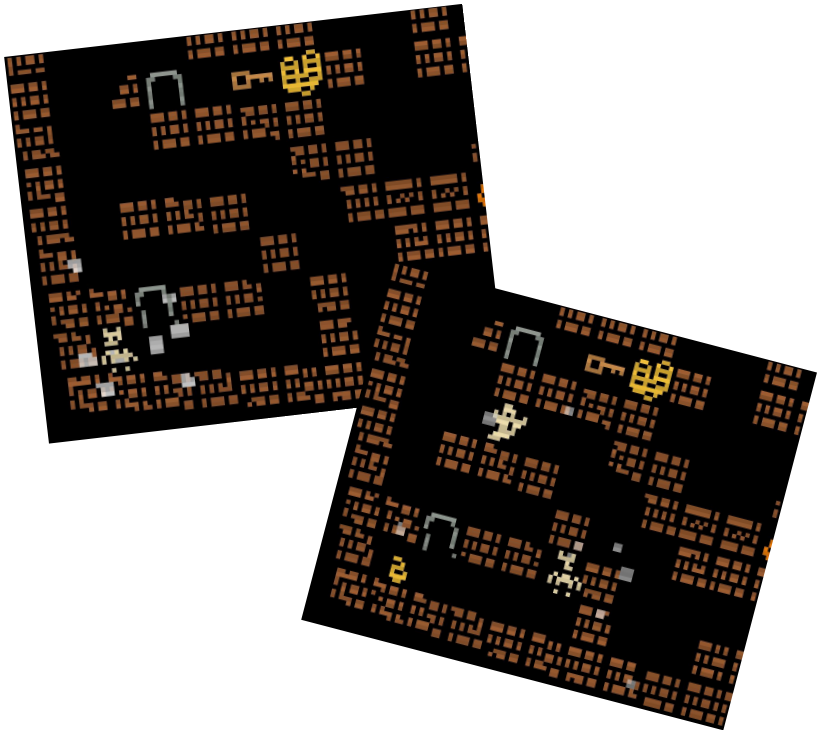


8. Level Constraints



9. Special effects

Particles



Particles are tiny graphical elements used to simulate complex visual effects such as **fire**, **smoke**, **sparks**, **rain**, **magic, etc...**

Instead of manually animating every detail, a particle system spawns and controls **hundreds** of small sprites or points, each with its own properties like size, color, lifetime, and movement.

Particles: Examples

Minecraft

Portal

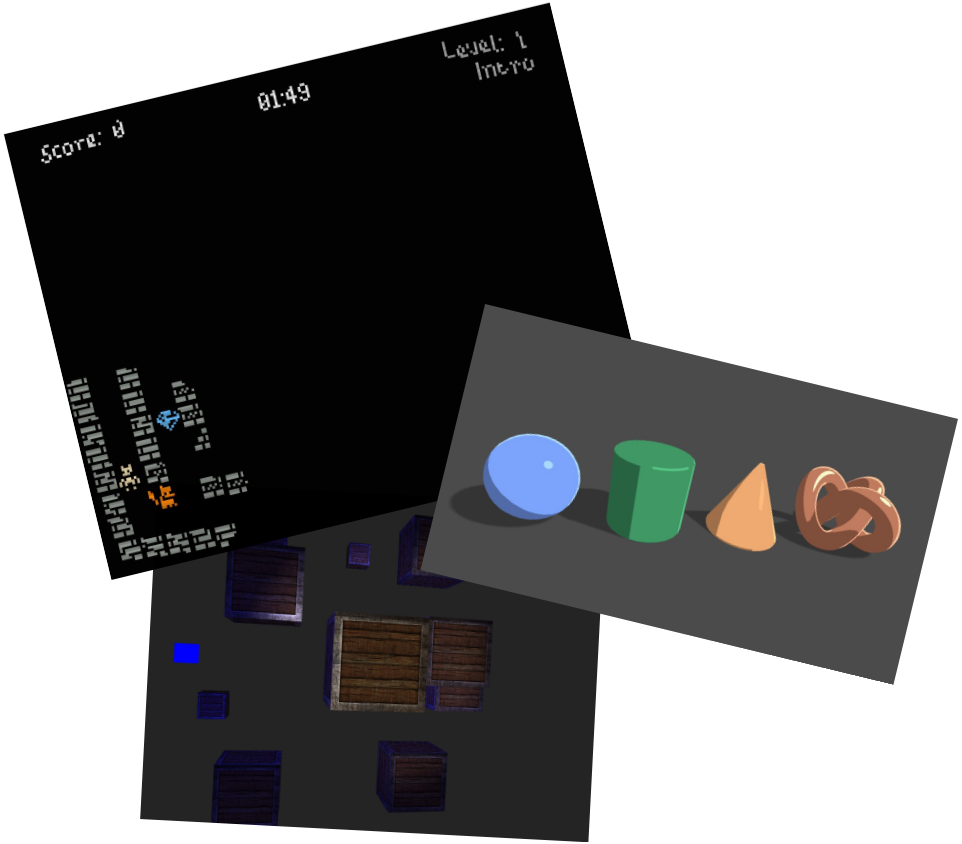
Pokemon Fire Red

Images courtesy of Mojang Studios, Valve Corporation, Nintendo Co. Ltd. (left to right)

Shaders

Shaders are small programs that run on the graphics card to control how objects are drawn on the screen.

Developers can create effects like: reflections, glowing surfaces, rippling water, or dramatic shadows.



Shaders: Examples

Jet Set Radio

Pokemon Fire Red

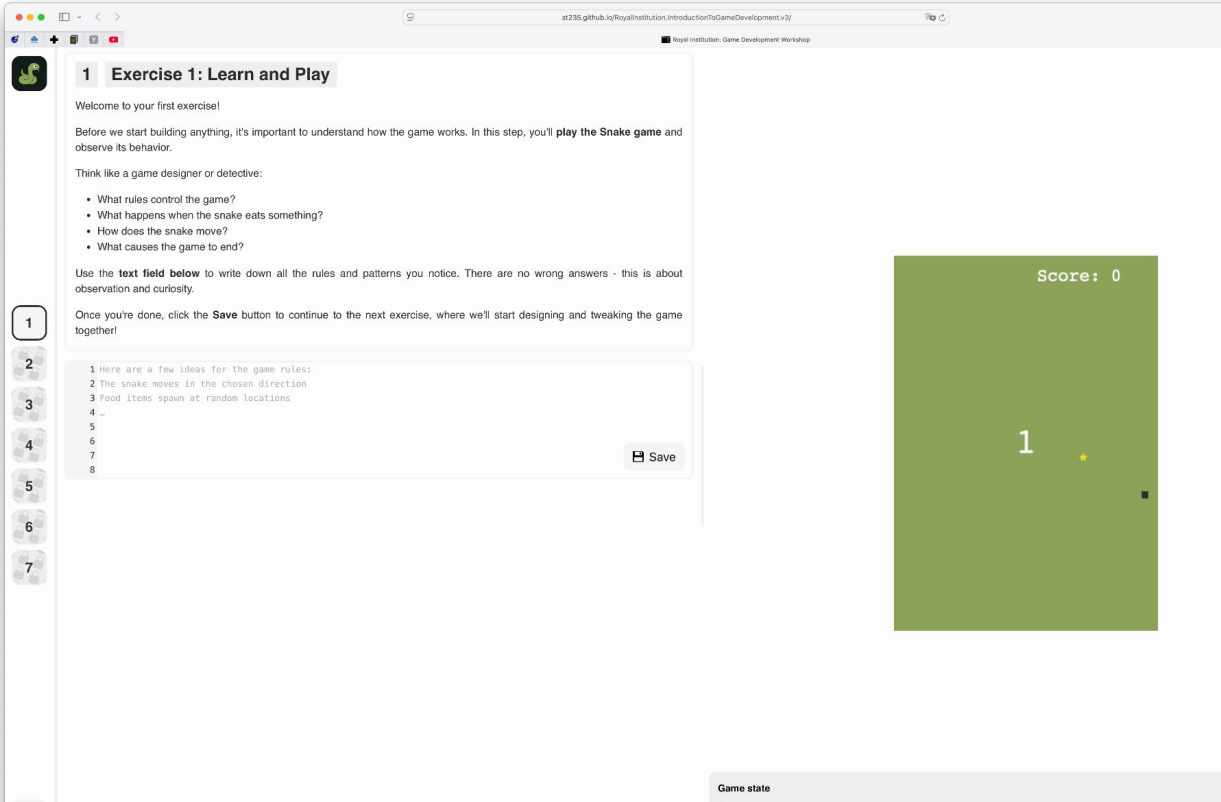
*Images courtesy of Sega Corporation, Nintendo Co. Ltd. (left to right)

Final thoughts and reflections

Explore other activities: Snake Game Design

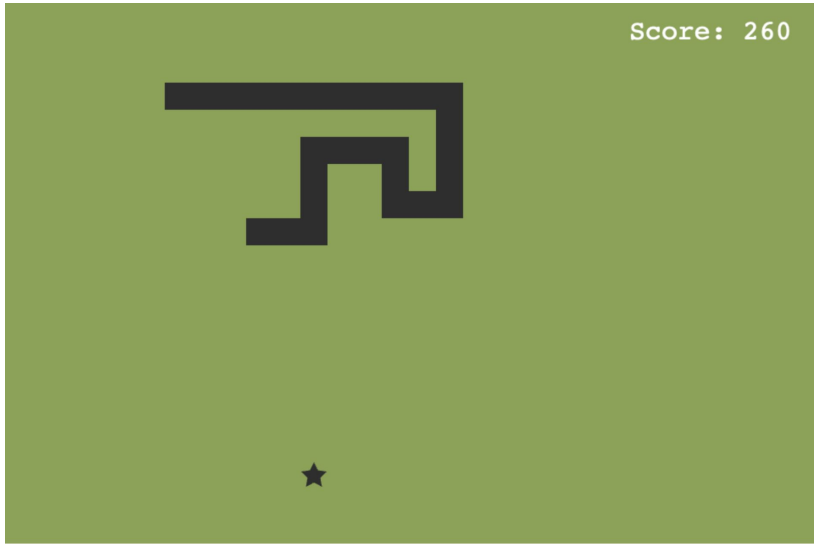
🔗 st235.github.io/RoyalInstitution.IntroductionToGameDevelopment.v3

Snake IDE



Explore other activities: Developing Snake Game

🔗 st235.github.io/RoyalInstitution.IntroductionToGameDevelopment.v2
🎥 https://docs.google.com/presentation/d/1Mjba1zi_cBy_Yj6Ie7shPmTOuNbAomML247daykqZMo/edit



Snake: Phaser Game

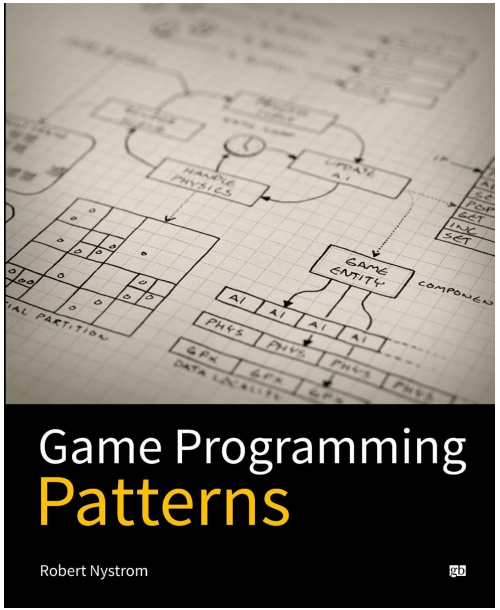
Snake: Slides

Further reading

“Game Programming Patterns” by Robert Nystrom

🔗 gameprogrammingpatterns.com

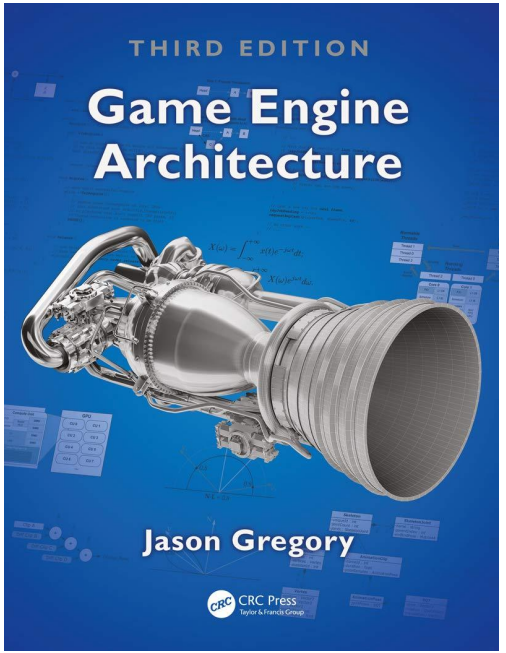
Game Programming Patterns is a collection of patterns and advice the author found in games that make code cleaner, easier to understand, and faster.



“Game Engine Architecture” by Jason Gregory

🔗 gameenginebook.com

Game Engine Architecture covers both the theory and practice of game engine software development, bringing together complete coverage of a wide range of topics. The concepts and techniques described are the actual ones used by real game studios like Electronic Arts and Naughty Dog.



CS50G: Introduction to game development

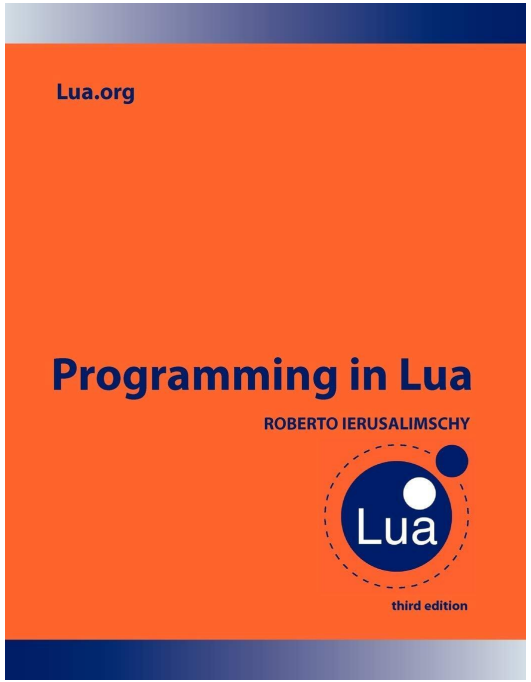
🔗 cs50.harvard.edu/games/2018/

Students explore the design of such childhood games as Super Mario Bros., Legend of Zelda, and Portal in a quest to understand how video games themselves are implemented. Via lectures and hands-on projects, the course explores principles of 2D and 3D graphics, animation, sound, and collision detection using frameworks like Unity and LÖVE 2D, as well as languages like Lua and C#.

Lua programming language

🔗 lua.org

Lua is a powerful, efficient, lightweight, embeddable scripting language. It supports several programming styles: procedural, object-oriented, functional, data-driven, and data description.



Phaser

🔗 phaser.io

Phaser is a free, open-source **JavaScript** framework for making 2D games that run in a web browser. It's especially popular for **prototyping**.

**Image is a courtesy of Phaser Studio Inc*

Thank you for attention!

Alex Dadukin

Software Engineer

*For any inquiries or questions contact Ben and he will kindly redirect them to me