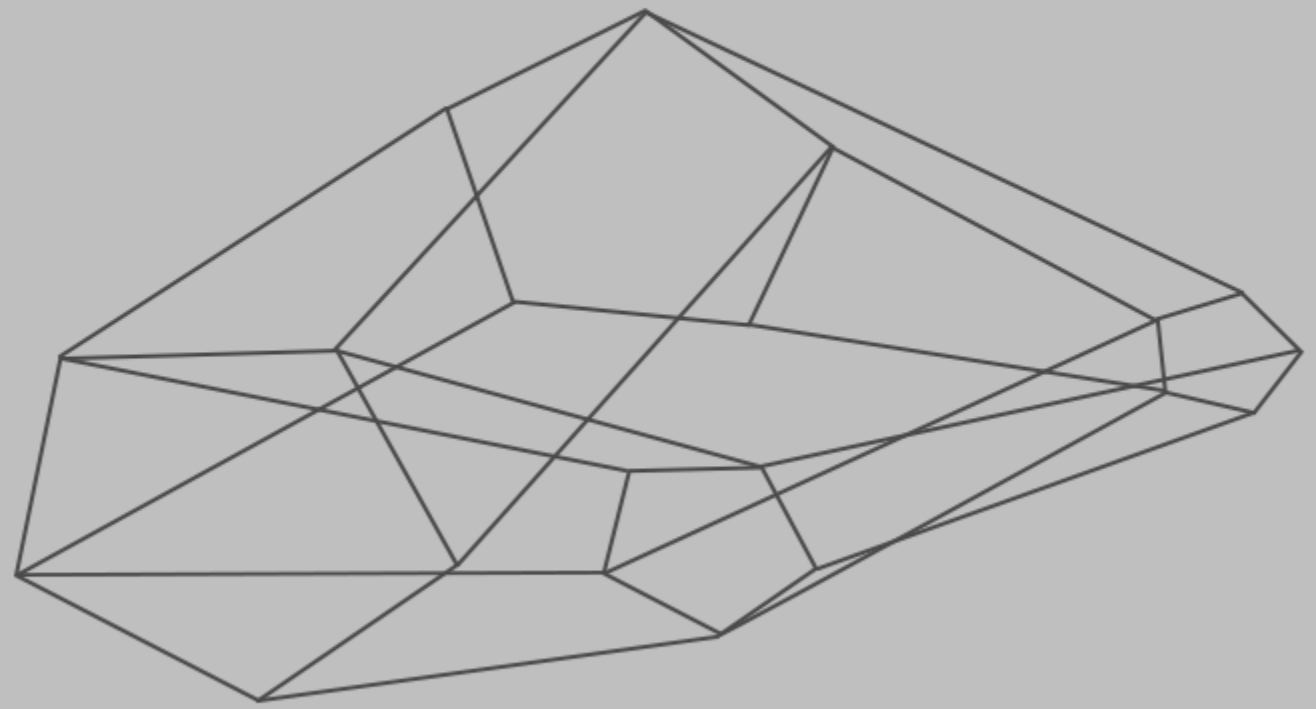
So, you want to make a game.

Loading...

// So, you want to make a game.

: : A talk on the basics of
creating your first videogame.
: : : : Concepts
: : : : Design
: : : : Technical
: : : : Artwork



Who am I?

- Hello!!
- I'm a 'Technical Designer'
- Post-graduate at Staffordshire University
- Big love for arcade & horror games, and nature
- Also make music and art sometimes!

nest
research & development

Some of my videos!

Who is this talk for?

Newcomers

+

Veterans

Who is this talk for?

What isn't this about?


VENNGAGE

JOB OPENINGS

The Venngage team is currently looking for candidates to fill the position of:

DEVELOPER

JOB DESCRIPTION:

A front-end developer specializes in building the front end, or client-side, of a web application, which encompasses everything that a client, or user, sees and interacts with.

REQUIRED SKILLS:

- Javascript
- CSS
- HTML

PROGRAM ASSISTANT

JOB DESCRIPTION:

A Program Assistant provides operational and administrative assistance to the Program Leader and Program Staff, performs a variety of administrative, coordination and logistical services in support of the operations of the Program, and assists with information management the team.

REQUIRED SKILLS:

- Organized
- Meets deadlines
- Multi-tasking

APPLY NOW!

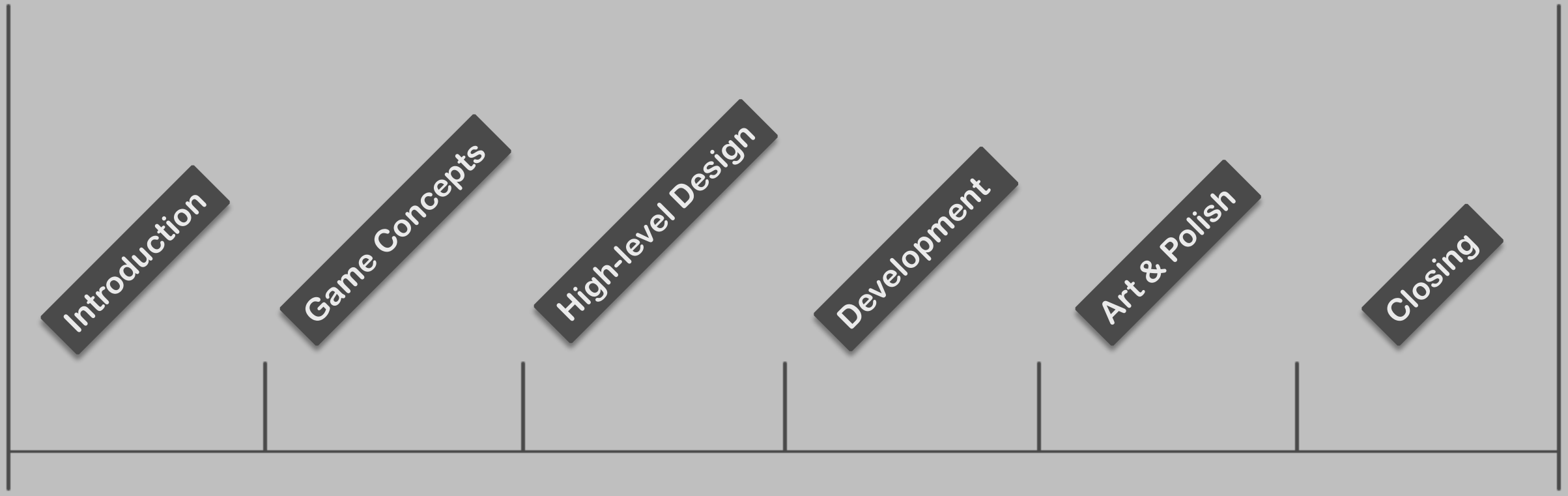
These are NOT real job opportunities at Venngage. This is just a template. Contact us at: venngage.com

Resumés and jobs

Information overload

Business/Marketing

// Contents


Introduction
Game Concepts
High-level Design
Development
Art & Polish
Closing

01 PART A Game Concepts

Planning, scope, research, defining your game.

Where do I start?

Game Essence Statement!

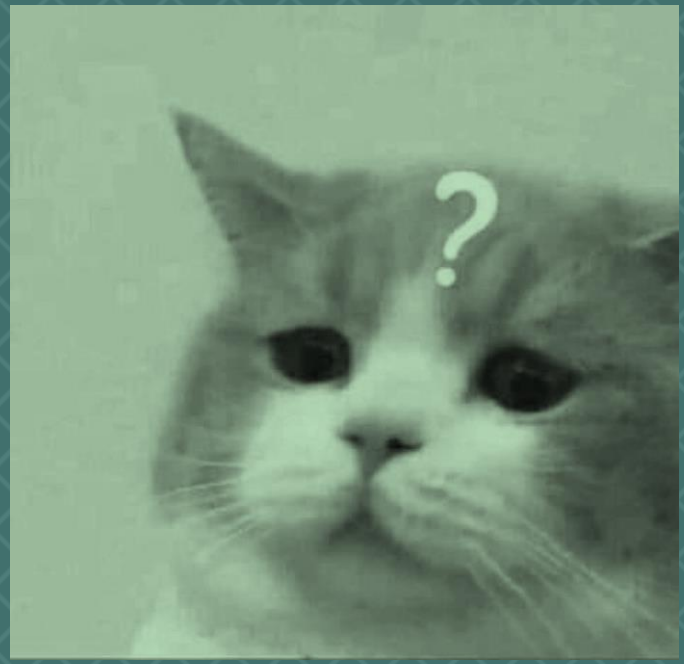
Examples of essence statements

*"It's time to leave home!
Get ready to move to the idyllic forest town of Animal Crossing, and settle down in the heart of the community, get to know and befriend your animal neighbors, share gifts, run errands, create your own outfits, decorate your new home - or maybe just take it easy at one of the many annual festivals!"*

“A father-daughter dynamic is tested by a highly infectious disease, heartbreaking sacrifices, and human cruelty. Explore, sneak and fight your way through the ruined city of Boston after a deadly pandemic in a desperate effort to save humanity."

Inspiration & research

- Start with simple keywords – ‘a vibe’.
- Sometimes you hit creative block, and that’s okay!
- Have a critical eye when researching.
- Analyzing games is a skill that improves with time.



Inspiration outside of games

- Unironically, touching grass and going out into the world does wonders for inspiration.
- Look across all media, not only games.
- Take up a secondary hobby to game development or use an existing hobby as a source to tap into.
- Base your work on old memories (autobiographical!).

"Chiba is … quite a distance from Kyoto, and when I moved there, I left my family and friends behind. In doing so, I realized that being close to them – being able to spend time with them, talk to them, play with them – was such a great, important thing. "

- Katsuya Eguchi

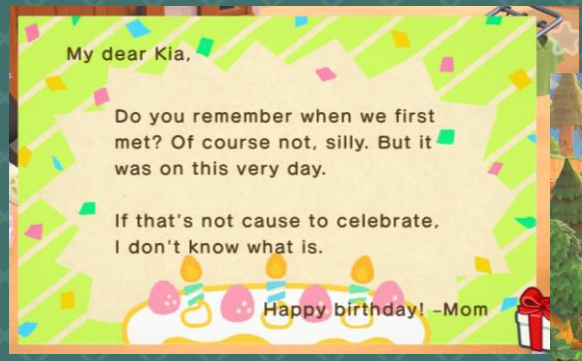
What is the experience you are trying to create for your players?

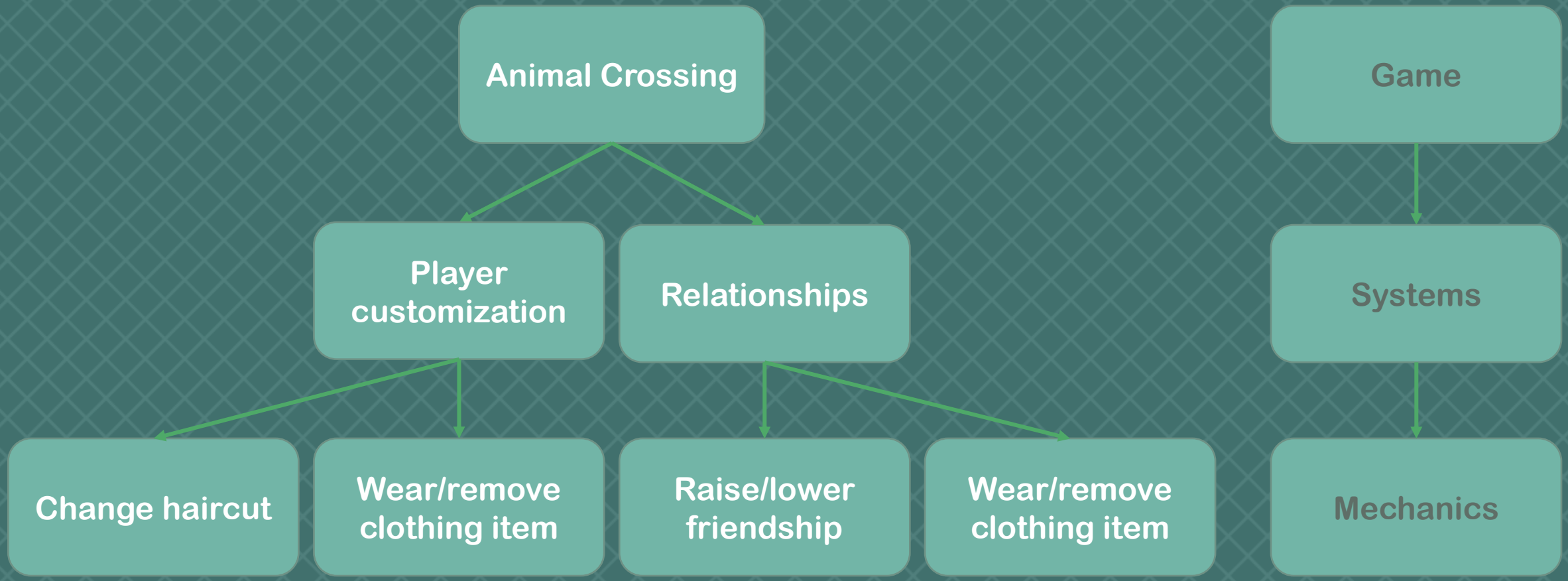
Formulating plans from essence statements

Dissected essence statement > Systems & Mechanics

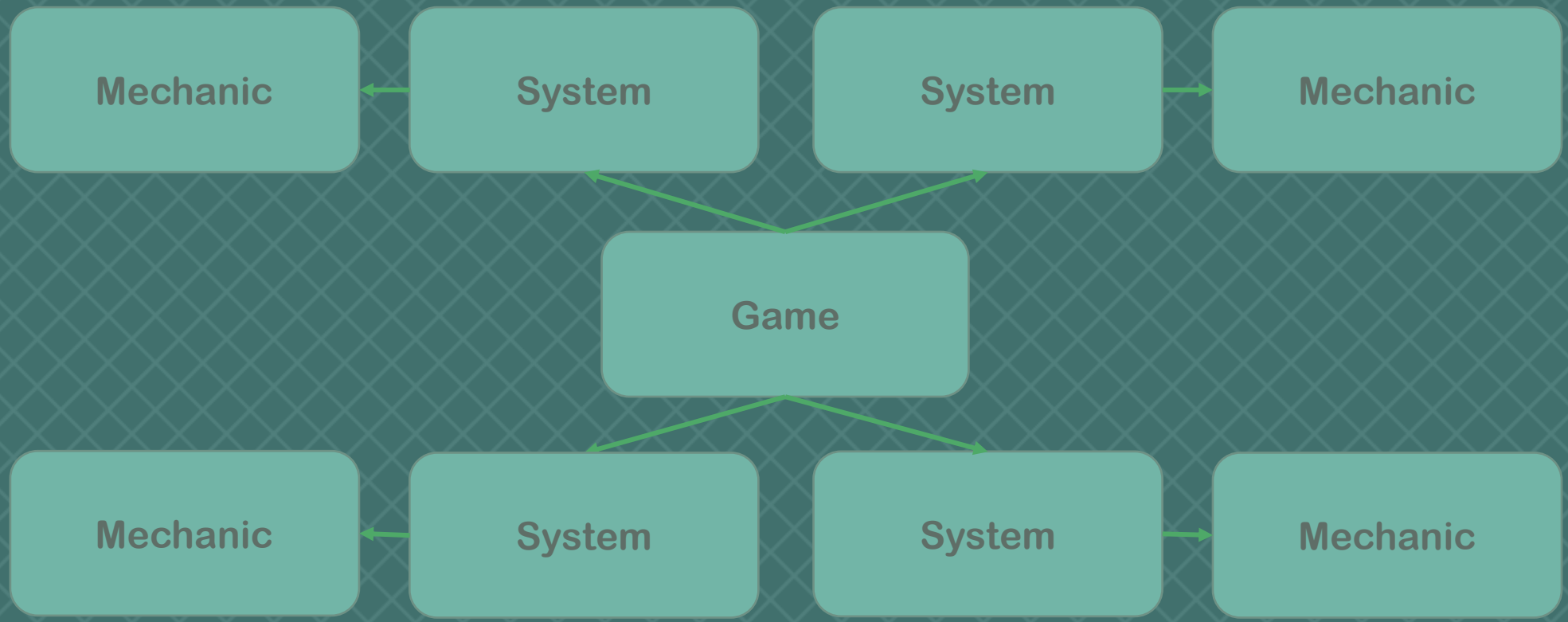
"It's time to leave home!
Get ready to move to the idyllic forest town of Animal Crossing, and settle down in the heart of the community, get to know and befriend your animal neighbors, share gifts, run errands, create your own outfits, decorate your new home - or maybe just take it easy at one of the many annual festivals!"

>




Animal Crossing
Player customization
Relationships
Change haircut
Wear/remove clothing item
Raise/lower friendship
Wear/remove clothing item
Game
Systems
Mechanics

Defining core systems & mechanics



Detailing mechanics

Hub Area

Inspired by The Legend of Zelda: The Minish Cap's hub world, and the branching design of Metroidvania games, the hub area offers everything the player needs to prepare before they enter the Underground Areas. The hub area starts off as one isolated area with little to explore at the beginning of the game, but homes become accessible and the explorable area eventually expands as the player rescues important story characters. Paths on the borders of the hub area lead out to mountain caves, which are explored routinely throughout the game.

Features

- **The Inn** – Save point, where NPCs can gather. Vital story progression moments happen here.
- **Apothecary** – Grants the player healing cards once rescued. Can level up specific cards if given special trade items.
- **Flower shop** – Respawn point if the player character is ever knocked out.

Mechanics

- **Access caves** – Exiting off the side of the screen from any exit point opens a prompt asking if the player wants to explore a cave. If they answer yes, a list of caves to travel to will be displayed. If the player answers no, the player will be forcibly walked back into the hub area.
- **Upgrade cards** – By communicating with rescued villagers, fulfilling requests from the Inn, or trading special items with the Apothecary, new versions of cards will become available to add to the player's deck of cards.

>

Scope and how to keep a project manageable

- Be wary of scope/feature creep!!
- Design by subtraction

"Perfection is achieved, not when there is nothing more to add, but when there is nothing left to take away."

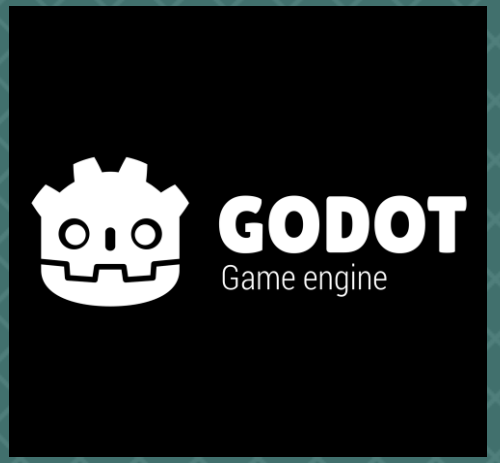
- Antoine de Saint-Exupéry

MoSCoW lists

MoSCoW

Must Have	Should Have	Could Have	Won't Have
1. Advanced player movement (parkour) 2. Basic combat mechanics (weapons) 3. Inventory menu 4. A single enemy type 5. Small level to explore 6. Interaction mechanics 7. Atmospheric lighting 8. Full game loop 9. Basic health system	1. RPG player stats 2. Upgrade menu 3. C++ implementations 4. Three enemy types 5. One large level to explore 6. Puzzle mechanics 7. Health system using RPG player stats for defence and etc.	1. In-game currency 2. Wearable/equipment 3. Two levels to explore 4. Self-created assets (texture, mesh, audio) 5. Basic cutscenes 6. Env. storytelling 7. Dynamic world state 8. Randomizing puzzle solutions	1. A commercial-length game experience 2. Complex cutscenes 3. Friendly NPCs 4. More than two weapon types 5. A focus on physics 6. Complex C++ content 7. Powerful weapons

Game engines: which is right for you?



Designer orientated

Programmer orientated

Section end

1. Ideation and inspiration
2. Research
3. Essence statement
4. Identify systems and mechanics
5. Detail mechanics
6. Rescope and game project bounds
7. Choose a game engine

Resources for game concepting

- Masahiro Sakurai on Creating Games – Planning & Game Design playlist https://www.youtube.com/playlist?list=PLgKCjZ2WsVLSmJwGC258l2HhWgf-P877B
- Masahiro Sakurai on Creating Games – Game Concepts playlist
- https://www.youtube.com/playlist?list=PLgKCjZ2WsVLTbEJdN9Rs6FWnOjllj4Tdt
- Game Maker's Toolkit – How to think like a game designer https://www.youtube.com/watch?v=ilOIT3dCy5w
- Infallible Code – Project planning tips for game developers https://www.youtube.com/watch?v=8m859pxcyLY
- Videogame Workshop – Core Game Mechanics https://www.videogameworkshop.com/game-design/Core-Game-Mechanics.html
- Game Developer https://www.gamedeveloper.com/

0 // PART 3 Game Design

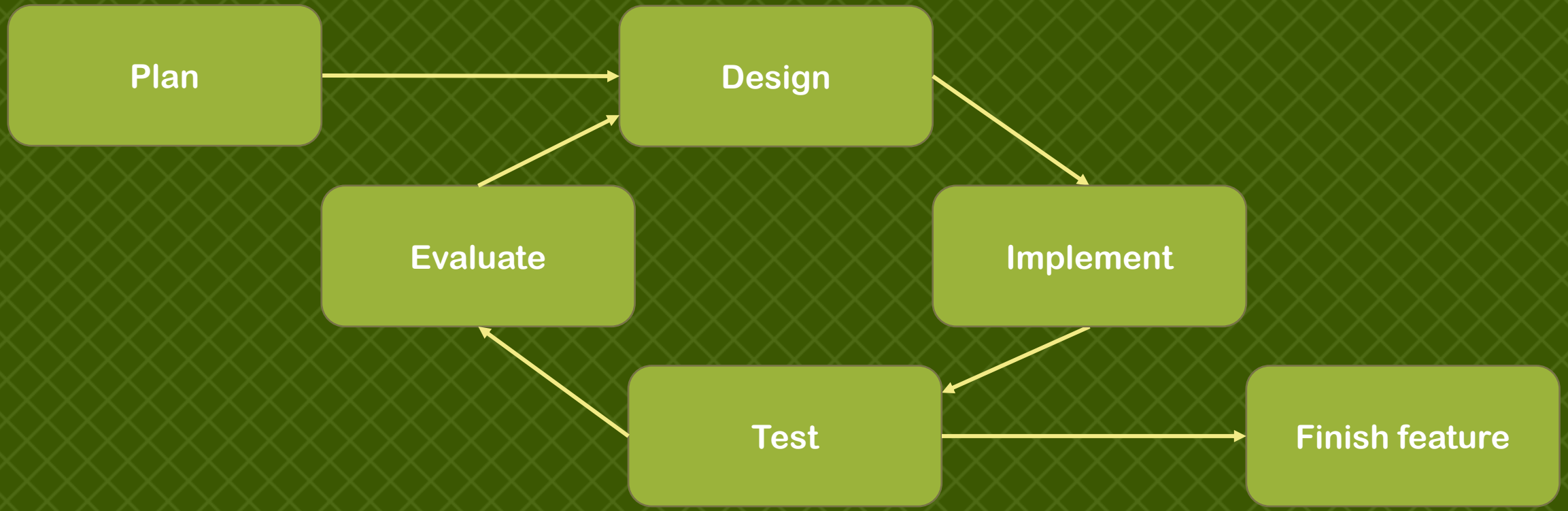
Specifications, 3Cs, UI, puzzles & level design.

What's next?

What does a designer do?

So, you want to make a game.

Iterative cycle



Silent Hill Remake Documentation

Contents

Table of Contents
-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-

==========

1) PROLOGUE

==========

(Taken from the Instruction Booklet)

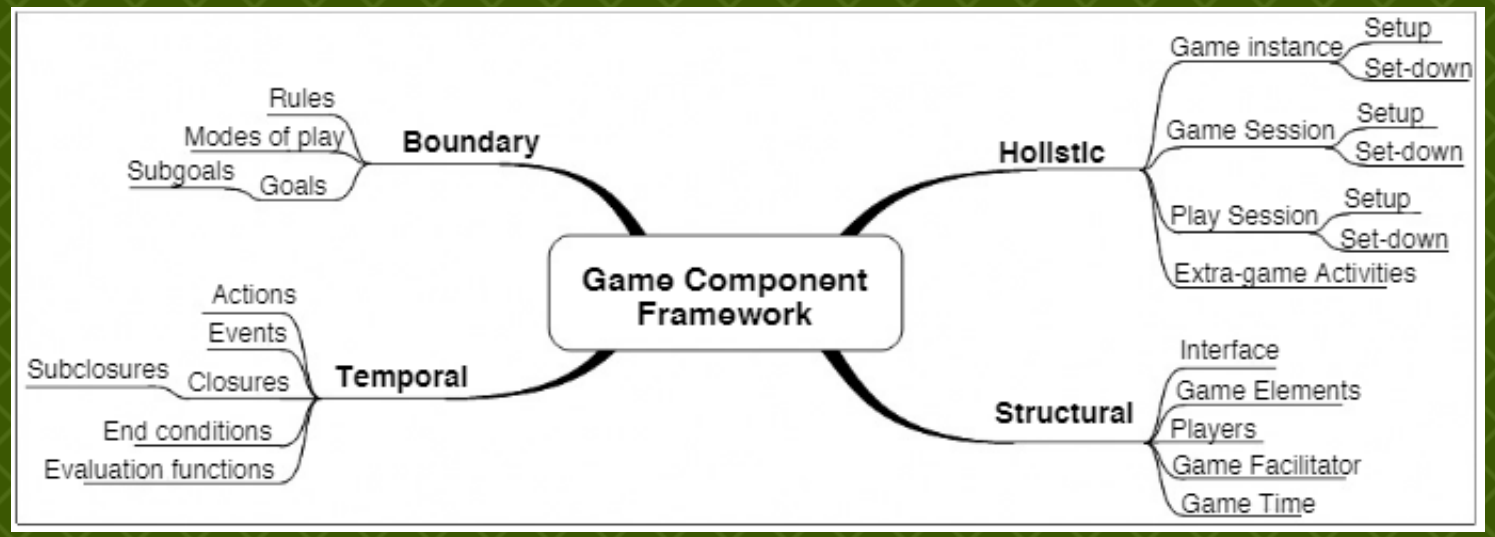
The Chozo.. Over millennia, this bird-like race of creatures made incredible technological and scientific leaps. Travelling at will through space, they built many marvel across the universe - technological wonders of unfathomable complexity and cities unmatched in beauty. They shared knowledge freely with more primitive cultures and learned to respect and care for life in all its forms.

Even as their society reached its technological peak, however, the Chozo felt their spirituality wane. Their culture was steeped in prophecy and lore, and they foresaw the decline of the Chozo coinciding with the rise of evil. Horrified by the increasing violence in the universe, they began to withdraw into themselves, forgoing technology in favour of simplicity. Tallon IV was one of several refuges and was they built - a colony bereft of technology, built of natural materials and wedded to the land and its creatures.

The years passed, and in time a great meteor crashed into Tallon IV, sending massive a massive spume of matter into the atmosphere and impregnating the land with a cancerous element known as Phazon. This element immediately sank into the earth and water, poisoning life wherever it bloomed. Most plants and animals died, while others mutated into hideous forms.

Design frameworks

- MDA (Mechanics, Dynamics, Aesthetic)
- DDE (Design, Dynamics, Experience)
- Octalysis Framework
- Björk and Holopainen's Activity-based Framework



Design frameworks

Design

Design/Björk and Holopaienen Design Model

Using Björk and Holopaienen's design model/framework, the components of the game exclusively from a design perspective are outlined below. Describing the game's patterns and structures in this way should assist in the development of the project and ensuring that it maintains its consistency.

Boundary

Rules:

1. The player moves at a slow pace and cannot run. Turn speeds are somewhat slow to help sell the possibility that something is behind the player.
2. Lighting and sound are vital to immersion.
3. Logic puzzles will be used to engage the player. They should provoke deep thought to keep them occupied.
4. Jumpscares will not be included. Jumpscares can ruin the experience for many. The dynamic tension system should contribute enough to unsettling the player appropriately.
5. Story should be interesting enough to encourage the player to keep going. Tell story via environment.
6. Player will not be able to see their avatar. Inability to connect with an avatar allows the player to be their avatar entirely.
7. The illusion of danger. A player should not be able to die but should feel that they are being jeopardized by the game's scenarios.
8. No inventory screen. Items are held in front of the player and brought around with them as physical objects.
9. Make the horror uncanny. Things happening out of the player's line of sight, unusual sounds, unexpected behaviours.

Modes of play:

- Standard gameplay - Player can move and interact with the world.
- Cutscene - Cinematics or dialogue will be delivered to the player.
- Interaction - Point of interest interactions may 'freeze' the world to create a sense of remoteness during puzzles.

Goals and sub goals:

- Thoroughly explore the game world.
- Solve puzzles to gain access to new areas.
- Read notes that gives the narrative context.
- Finish the story.
- Uncover details in the world that might reframe the story (set dressing).

Holistic

Game instance:

- Setup - Player wants a horror experience that they can enjoy. Game starts, most recent save is loaded.
- Set-down - Player finishes the game's story or saves their progress and exits the game.
- Tension value - Increment or decrement this value dependent on the player's behaviour in-game.

Play session:

- Setup - Player launches game. Immediately put into gameplay, no/minimal menus. Options menu on first launch?
- Set-down - Game is played until player has 'had enough' and quits. Progress up to last major scenario is saved automatically.
- Player readjusted on every relaunch of the game. What puzzles do they need to solve? What last happened in the narrative?

Extra-game activities:

- Reading notes for narrative context.
- Playing the game for a second time. Player behaviour is expected to be different during subsequent playthroughs, and so their experience will be too.

Temporal

Actions:

- Walking.
- Crouching.
- Holding key items/objects.
- Solving puzzles.
- Opening doors.
- Reading notes.

Events:

- The game starts. Dialogue appears on-screen to engage the player, similar to the quotes that appear when loading a save in Resident Evil (PSX).
- Tension triggers caused by player overlap and behaviour. New triggers appear as narrative advances. Scenarios will be different depending on player behaviour.
- Doors open or close.
- Object is dropped on floor. Objects in the world have generic physics to allow player to move or interact with them.
- The game ends, delivers final cinematic or dialogue.

Closures:

- Key item given to player as confirmation that they completed a puzzle. Possible audio cue?
- A room is devoid of new points of interest, or the previous points of interest are resolved/completed.
- Ending the narrative. The player has finished the experience and can stop playing.

End conditions and evaluation functions:

- A room is 'cleared'.
- The player needs to use a key item/object in another area.
- The player finishes reading a note.
- All puzzles and points of interest are resolved/completed.

Structural

Interface:

- In-world diegetic UI - A radio can be tuned to different stations, a map on the office wall shows the layout of the area.
- Standard WASD-mouse or gamepad support for first-person.
- Display of notes as text on-screen, for legibility.

Game elements:

- Player character - A detective supporting a missing persons case at a motel.
- Notebook - Keeps note of current stage of narrative.
- AM radio - Used to keep track of signals that may contribute to narrative or puzzle solutions.
- Scenarios - Puzzles, actions and story beats that comprise the game experience.
- Hallways - Must create liminal space by using corners.
- Rooms - Varying functions, such as office, bedroom, bathroom, storage space.
- Point of interest - Interactable object or puzzle that contributes to narrative progression.
- Tension triggers - Events that happen near the player at specific points in time.

Players:

- Must solve puzzles.
- Explores the game world for clues.
- Not represented physically, disembodied.

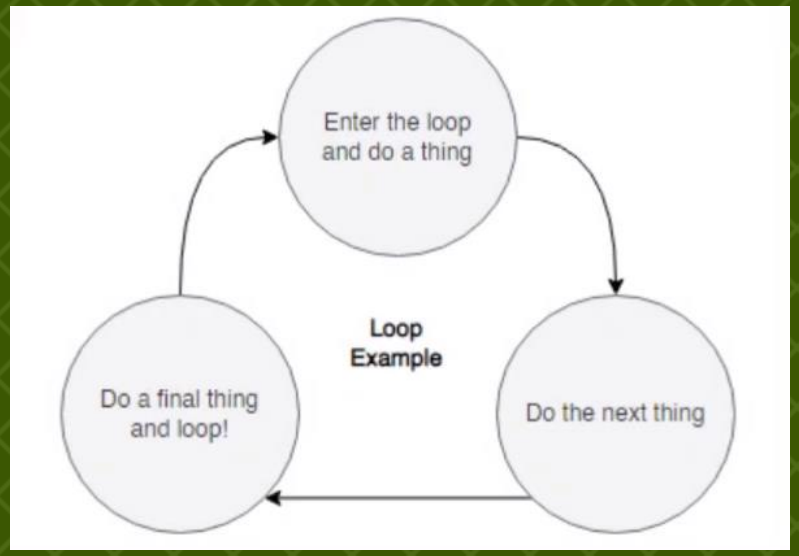
Game facilitator:

- Dynamic tension system - Tracks player behaviours and reacts accordingly. Affects game world.
- Game manager - Tracks player progress throughout narrative. List of game flags is changed per completion of each scenario.

Game time:

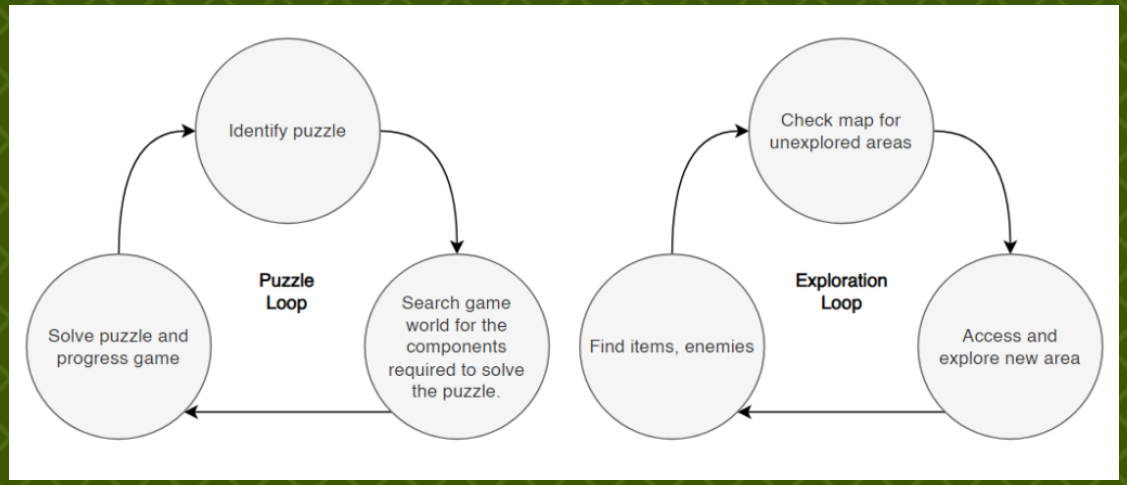
- Day and night change per scenario rather than in real-time.
- Tension activity stops while interacting with the notebook or a point of interest, to prevent frustration.
- Audio on the AM radio's frequencies changes as scenarios are completed.

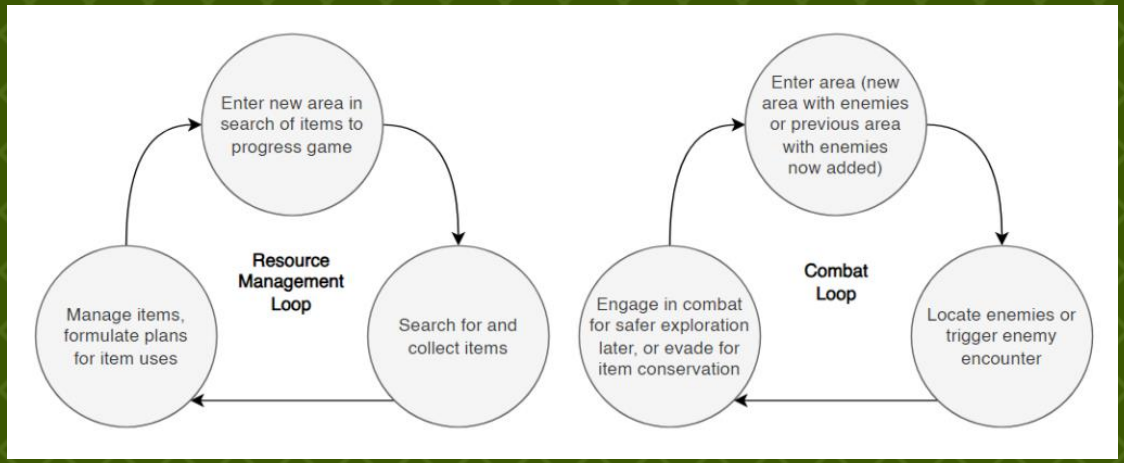
Gameplay loops

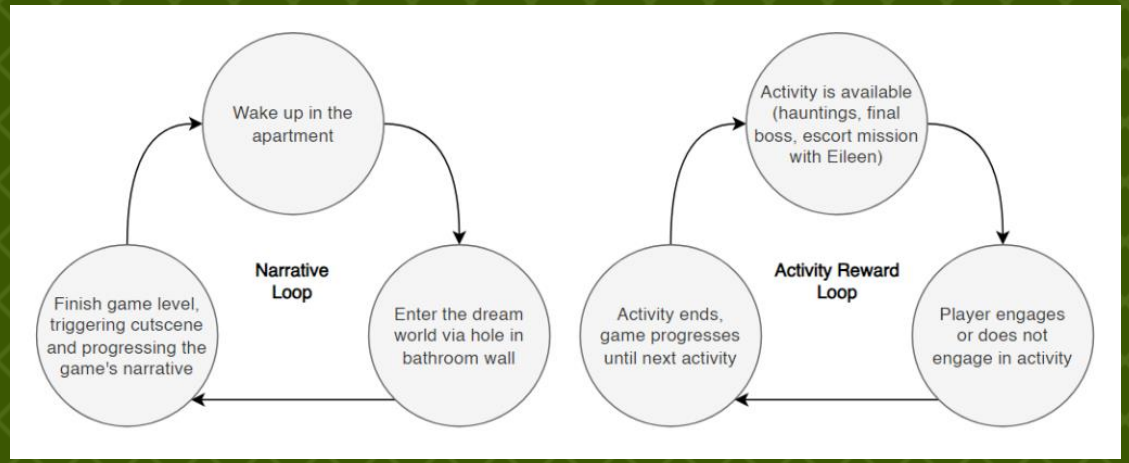


“actions undertaken by the player, most often the main actions defining the game”
- Bryan Wirtz, GameDesigning.org

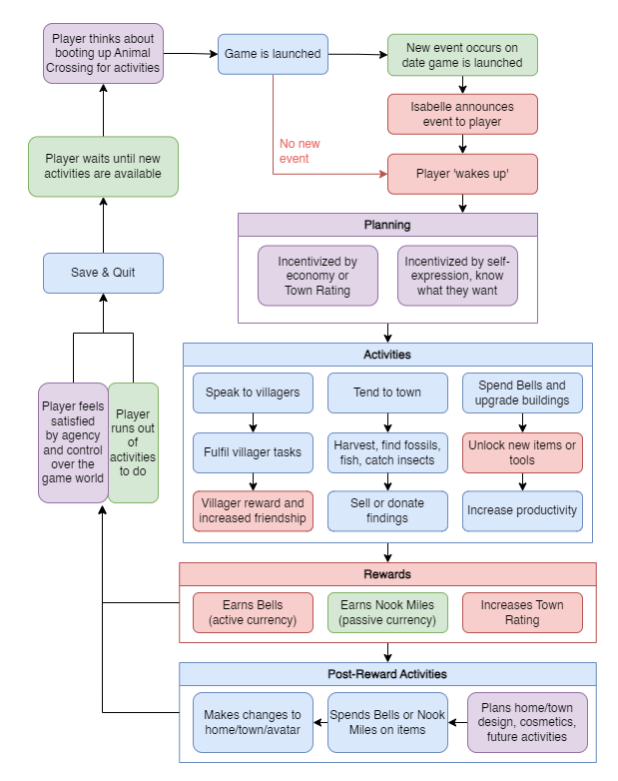
Identifying your gameplay loops

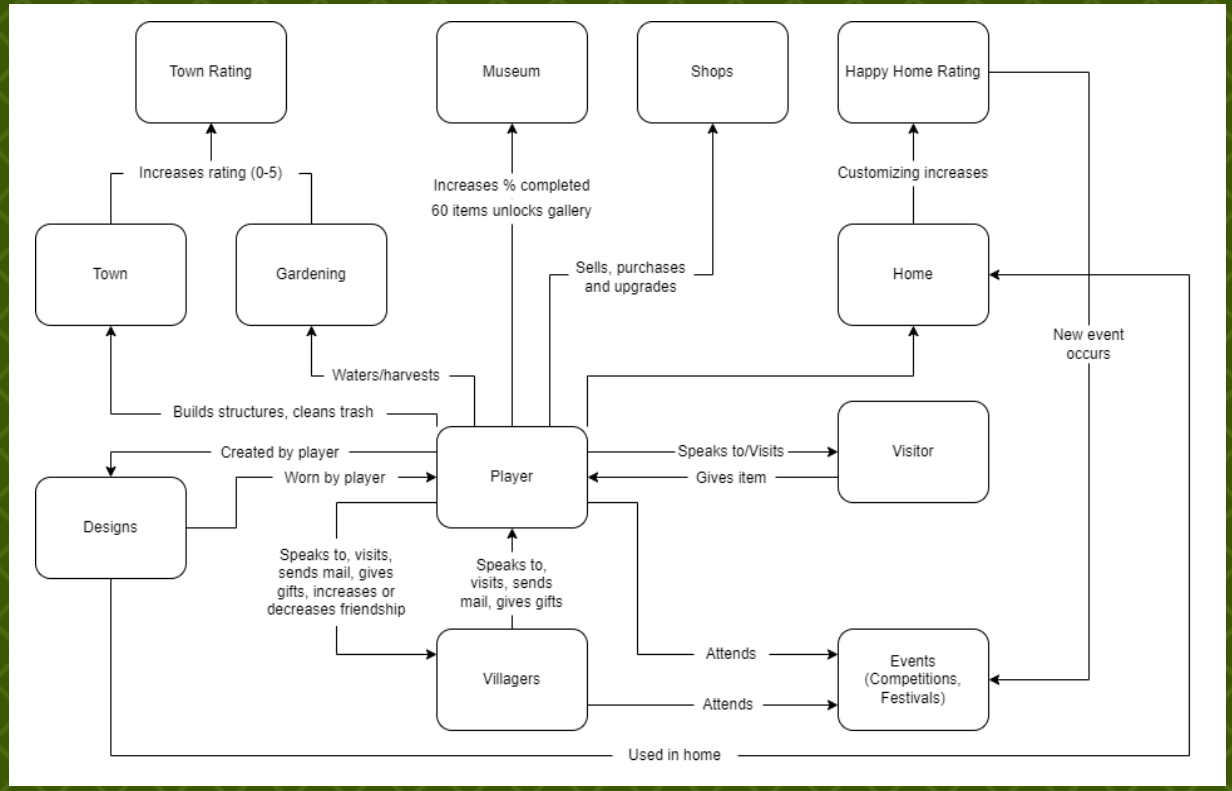




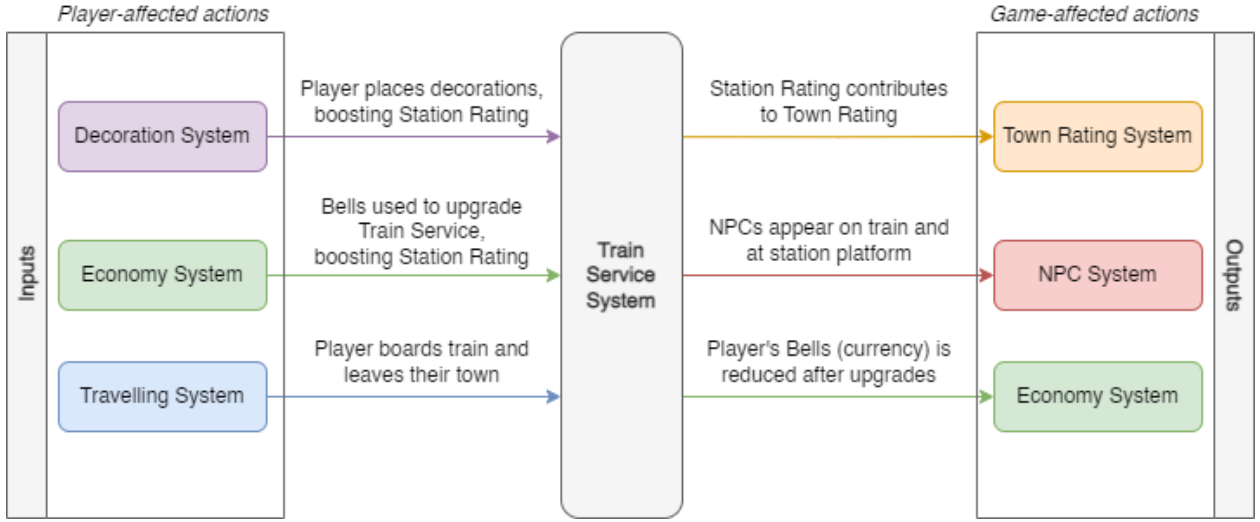


Systems design



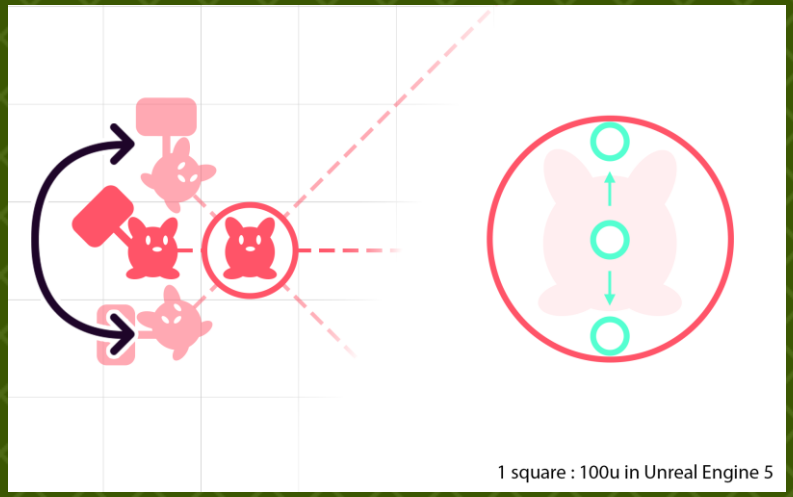


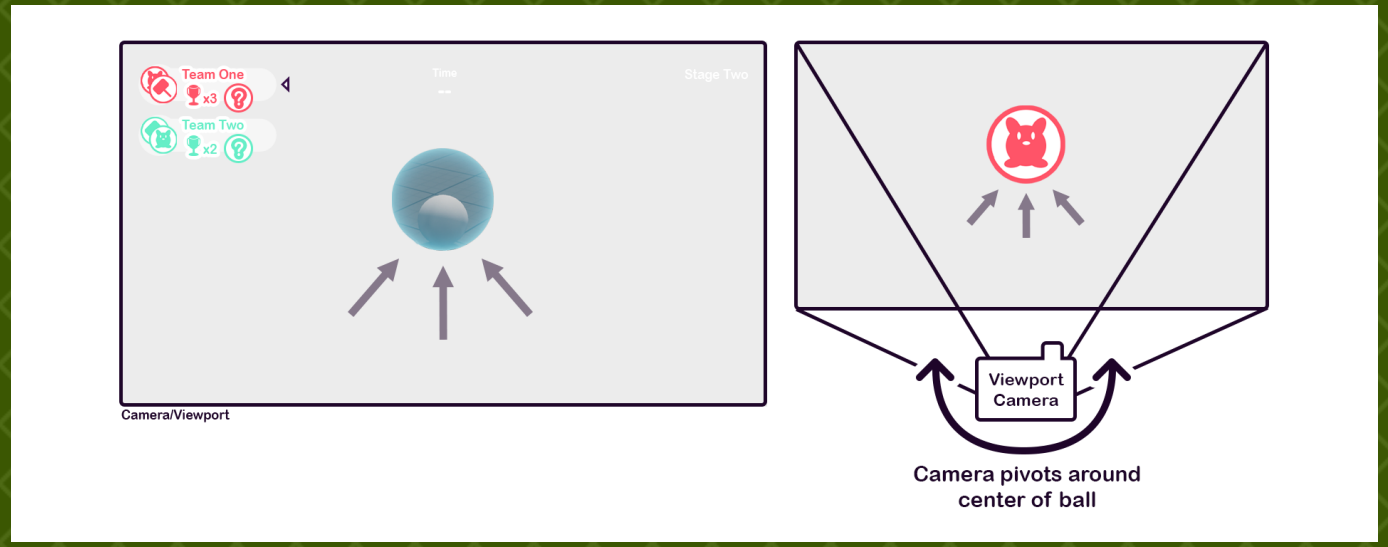
Systems relations

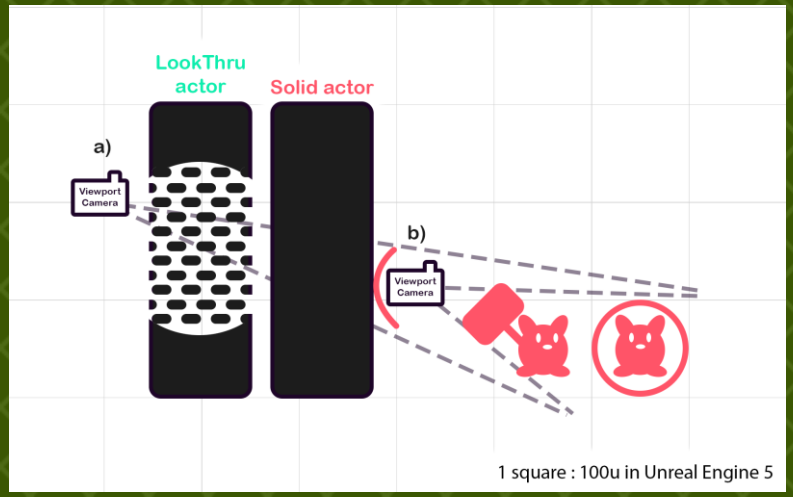


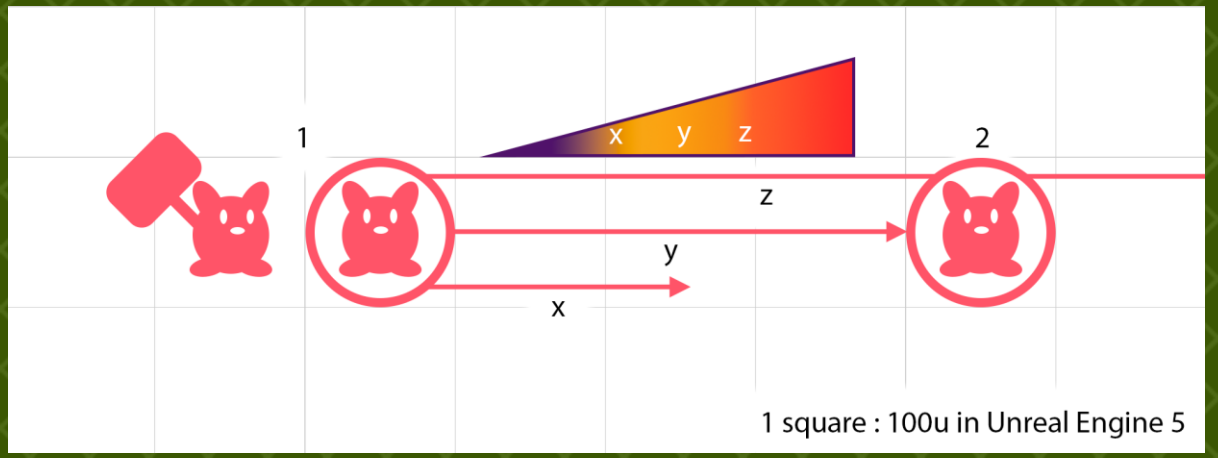
Diagrams created for free using www.drawio.com!

3Cs – Character, Controls, Camera

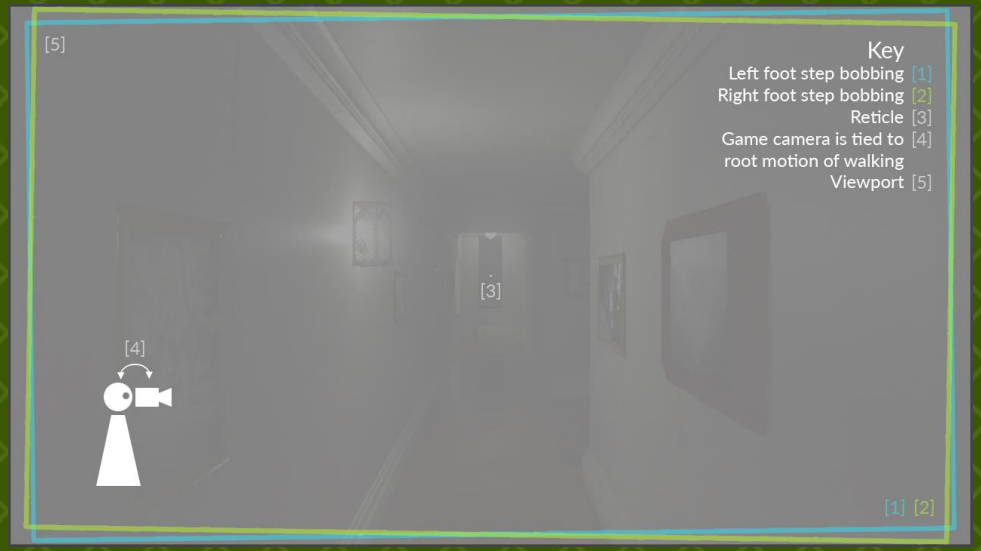








3Cs – Character, Controls, Camera

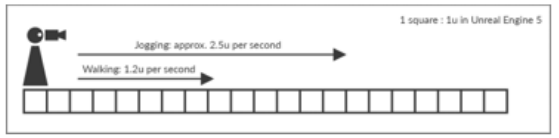


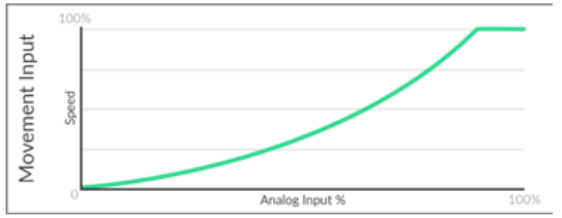
Character Statistics

	Default	Min	Max
Vitality / Health	*100%*	*0%*	*Unkillable (during sequences)*
Walk Speed	*N/A*	*Over deadzone*	*1.2u / second*
Jog Speed	*2.5u / second*	*N/A*	*N/A*
Run Duration	*N/A*	*0.2 seconds*	*10 seconds*
Crouch Speed	*0.1 seconds*	*N/A*	*N/A*
Base Attack Speed	*N/A*	*0.5 secs / attack (uncharged)*	*1.75 secs / attack (charged)*
Interact Range	*N/A*	*10u*	*75u*
Keys Held	*1*	*0*	*8*
Camera FOV	*90 degrees*	*115 degrees*	*60 degrees (Focus mode)*
Look / Turn Speed	*45*	*30*	*70 (when jogging)*
Capsule Component Height	*192u*	*142u (crouched)*	*192u*
World Gravity	*-2744*	*N/A*	*N/A*

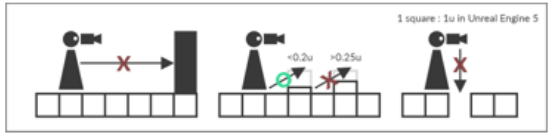
Movement & Physics

The player will move at maximum walk speed when the "Movement" binding is at >90% (analog and digital). From over the dead zone value and up to <90%, the player's speed increases on an upward curve, to give the player steadier control over their movement. When the "Run" binding is held, the player will increase their movement speed by 52%. Analog input no longer affects amount of speed.

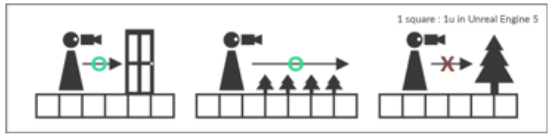




Objects and walls in the environment will block the player's movement. The player character's collider is a capsule, so that they pass smoothly around objects. Low platforms will be stepped up onto, while any over ¼ of a unit block movement. Similarly, holes in the floor block the player from passing over.



Walking into doors will automatically open them unless they are locked. Locked doors are treated similarly to wall collision. Small shrubs and foliage will overlap the player character collider but sway and create noise as they are moved through. Larger foliage, like trees, will entirely block the player and other colliders.



Non-diegetic (not literal)

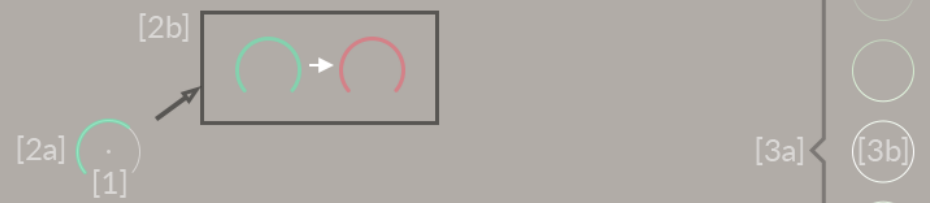
Diegetic (in game world)

UI and the Heads-Up Display

UI and the Heads-Up Display

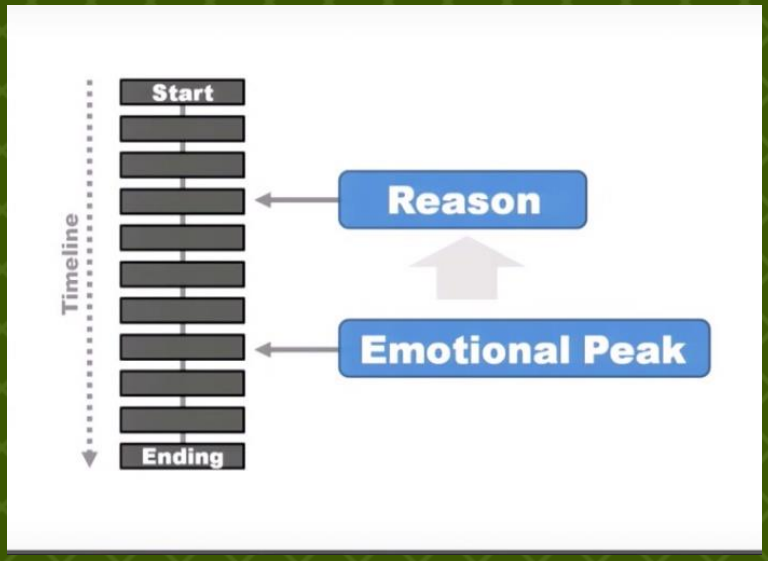
Key

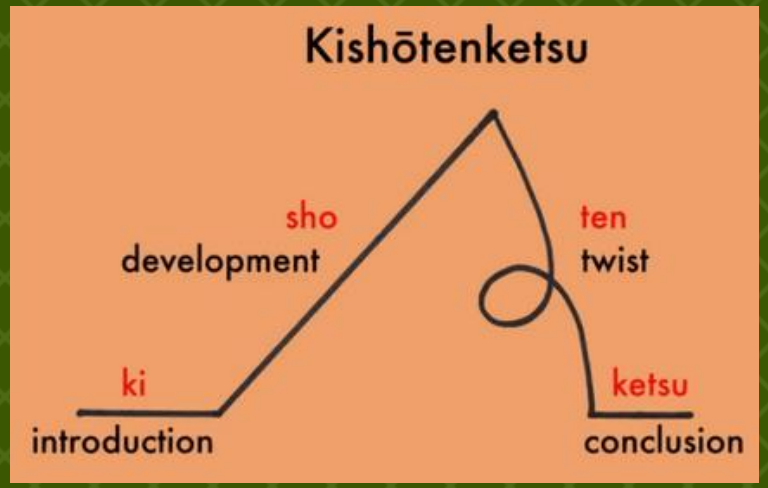
[1] Center recticle
[2] a) Current equipment status
b) Status change, aiming at enemy
[3] a) Equipment roster
b) Current equipment
[4] Subtitles and inspection text field
[5] Auto-save icon
[6] Viewport

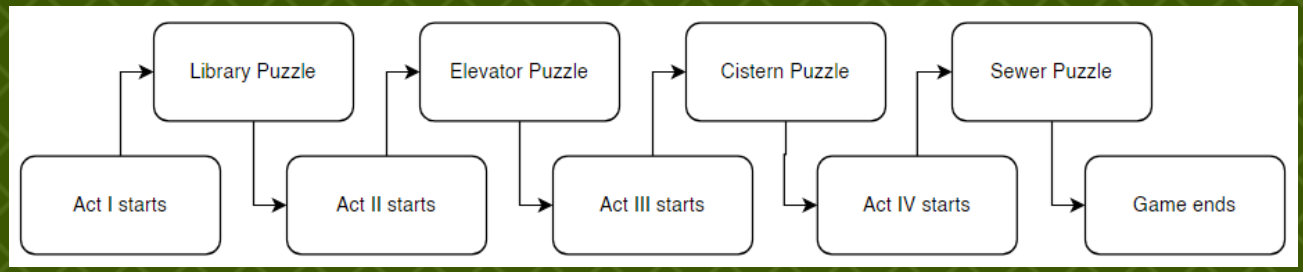



2017
TITLE LINE 01
2018
TITLE LINE 02
2019
TITLE LINE 03
2020
TITLE LINE 04
2021
TITLE LINE 05
2022
TITLE LINE 06


PATRICK HORVATH
PORTFOLIO EXCERPT: SHAUN OF THE DEAD
MCU OF CHECKOUT GIRL AS SHE FACES THEM
MCU SHAUN & ED "OH MY GOD! SHE'S SO DRUNK."
M.S. CAM TRACKS BACK W/ MOTION OF CHECKOUT GIRL AS SHE LURCHES
L.S. (LA) OF SHAUN AND ED. TRACK FORWARD W/ STAGGERING APPROACH


Start
Timeline
Reason
Emotional Peak
Ending

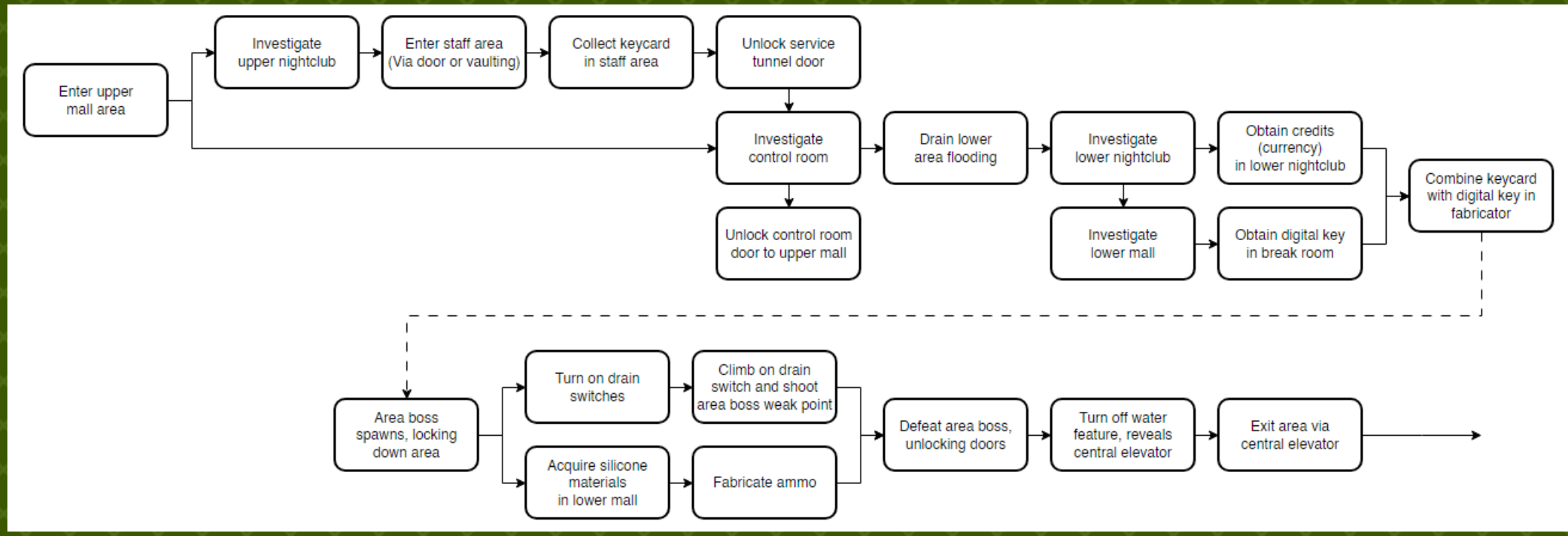

Kishōtenketsu
sho
development
ten
twist
ki
introduction
ketsu
conclusion


Library Puzzle
Elevator Puzzle
Cistern Puzzle
Sewer Puzzle
Act I starts
Act II starts
Act III starts
Act IV starts
Game ends

Scott Kim's Eight Steps of Puzzle Design:

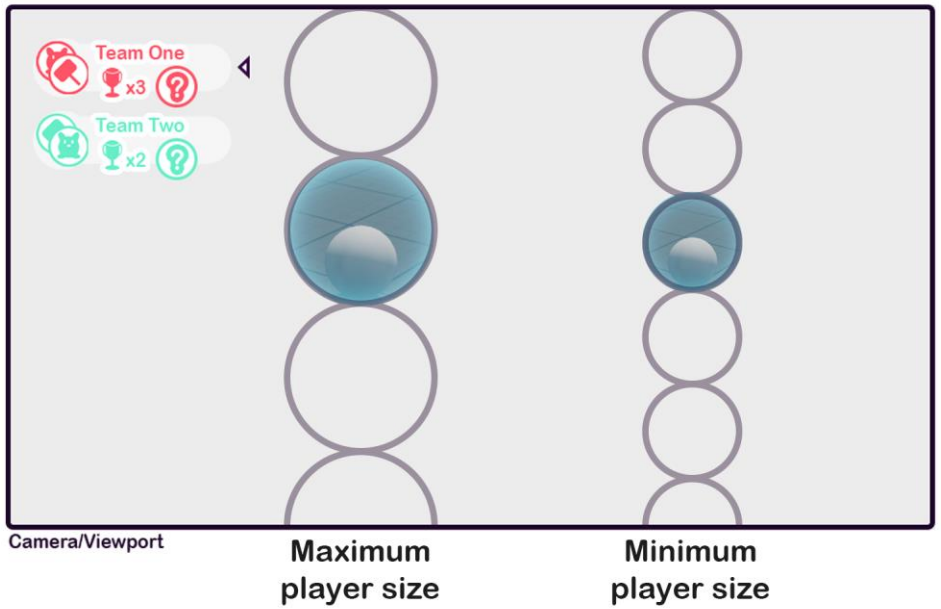
1. Finding inspiration from other sources,
2. Simplification of…
   a. the concept to one core player skill,
   b. detail, by eliminating irrelevant information,
   c. the play pieces used in the puzzle.
3. Creating a prototype to create and test puzzles,
4. Defining the rules (the game bounds, win/lose conditions, attributes),
5. Constructing puzzles in a way that…
   a. respects the logic of the game world,
   b. is a fair challenge,
   c. feels rewarding to players who complete them,
6. Testing for puzzle difficulty and errors in the game rules,
7. Planning the arrangement of puzzles and puzzle elements,
8. Polishing and presentation of the puzzles.

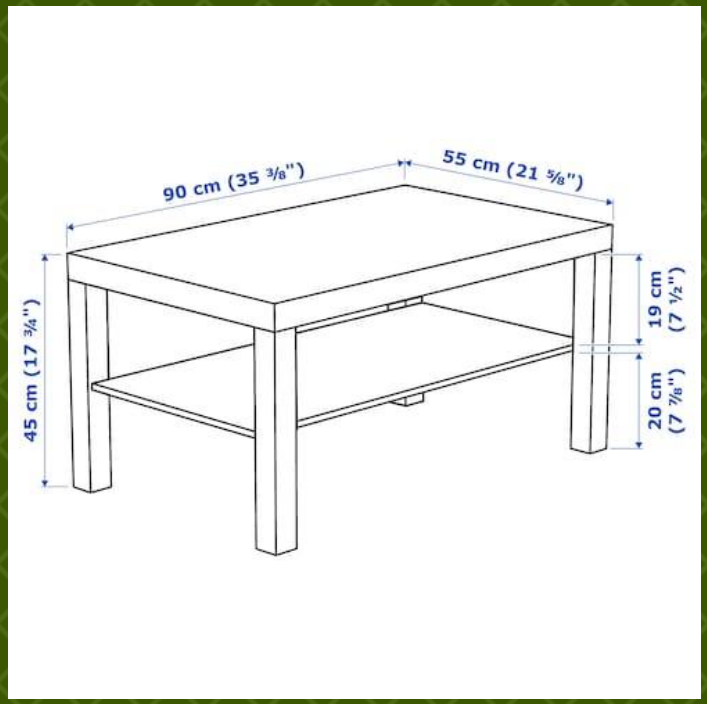
Puzzle design



Breadcrumbing and other means of affordance

Scale and reference



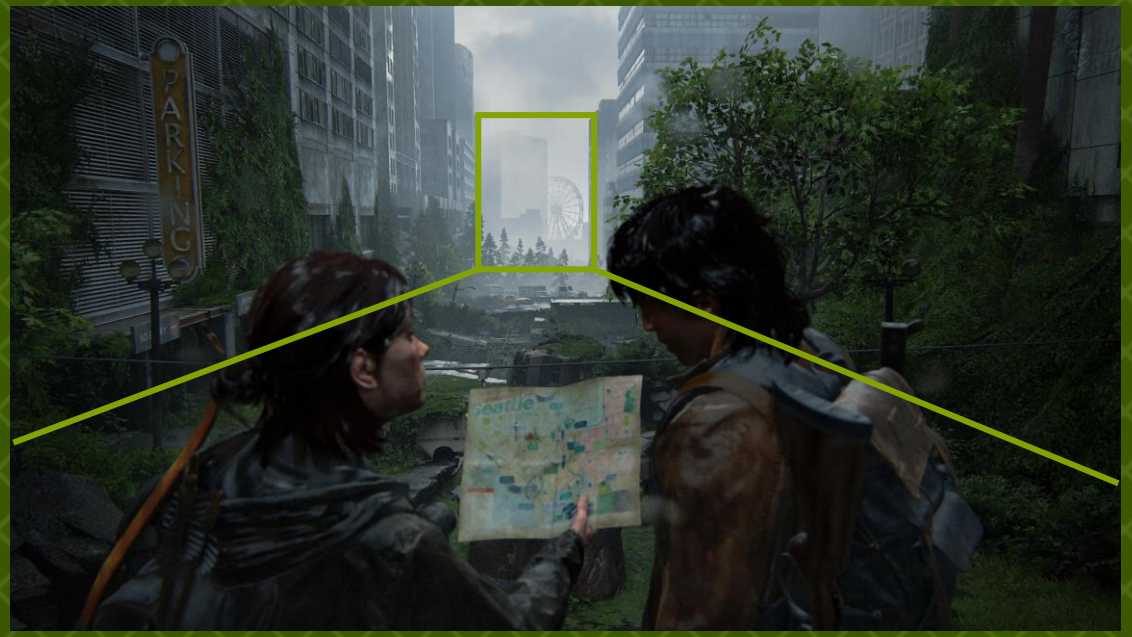


Blockouts

>

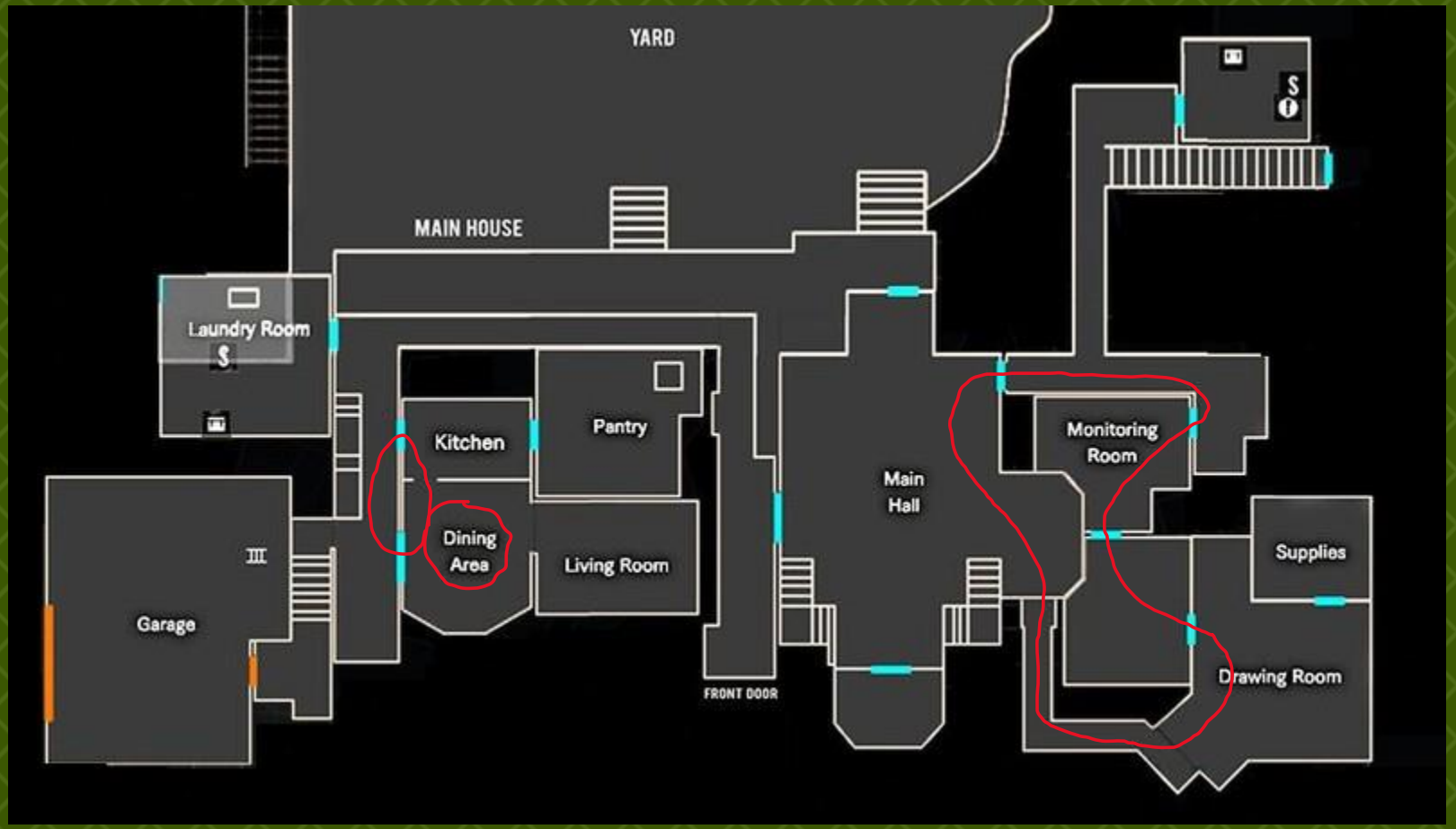
Set dressing

Focal points



Focal points

Loops and navigation variety



Loops and navigation variety

Section end

1. Iteration is persistent throughout design and development.
2. Begin a game design document from your initial plans.
3. Choose a design framework to analyze your project with.
4. Identify any gameplay loops.
5. Always consider the player's experience while designing.
6. Look at your game's design and systems holistically.
7. Expand on design specifications, be very detailed.
8. Plan your scenarios ahead of designing levels.
9. Affordance is vital to ensuring clarity for players.
10. Use scale reference as you work.
11. Consider focal points and leading lines.
12. Expect to do iterations on blockouts before art is added.
13. Keep level navigation interesting with loops.

Resources for game design

- “The Door Problem” of Game Design https://www.gamedeveloper.com/design/-quot-the-door-problem-quot-of-game-design
- Game Developer – How to write a game design document https://www.gamedeveloper.com/business/how-to-write-a-game-design-document
- Bjork and Holopainen – Patterns in Game Design https://www.researchgate.net/publication/236157130_Patterns_in_Game_Design
- GameDesigning.org – Understanding a Gameplay Loop https://www.gamedesigning.org/learn/game-loop/
- Ron Gilbert – Puzzle dependency charts https://grumpygamer.com/puzzle_dependency_charts
- World of Level Design – Blocktober Guide https://worldofleveldesign.com/categories/level_design_tutorials/guide-to-blocktober.php
- Kotaku – The Horror of Videogame Doors https://kotaku.com/the-horror-of-video-game-doors-1819921161
- Valve Developer Community – Level Design Loops https://developer.valvesoftware.com/wiki/Loops_(level_design)

0 // PART C Development

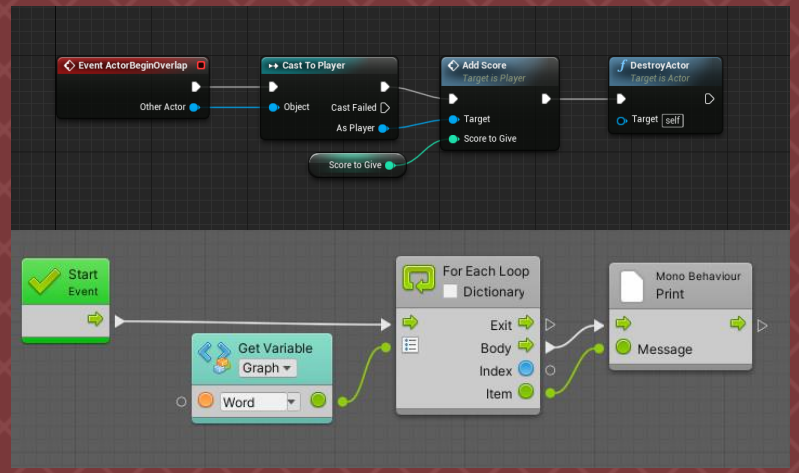
Prototyping,

Section start

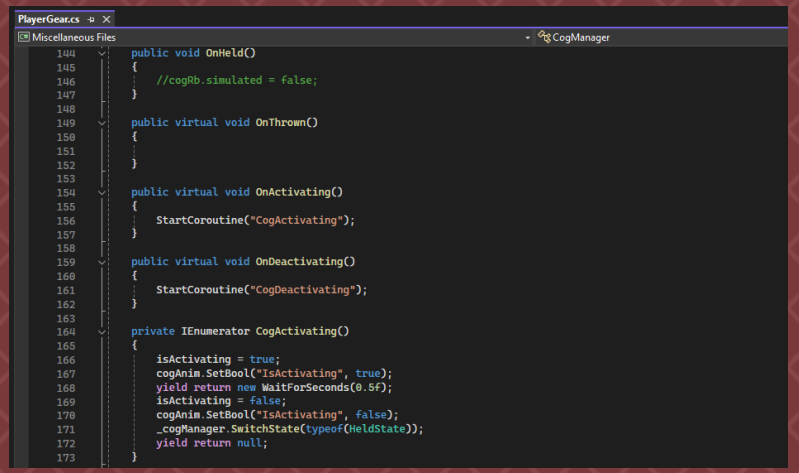
So, you want to make a game.

Prototyping helps to work efficiently
by failing often in small ways.

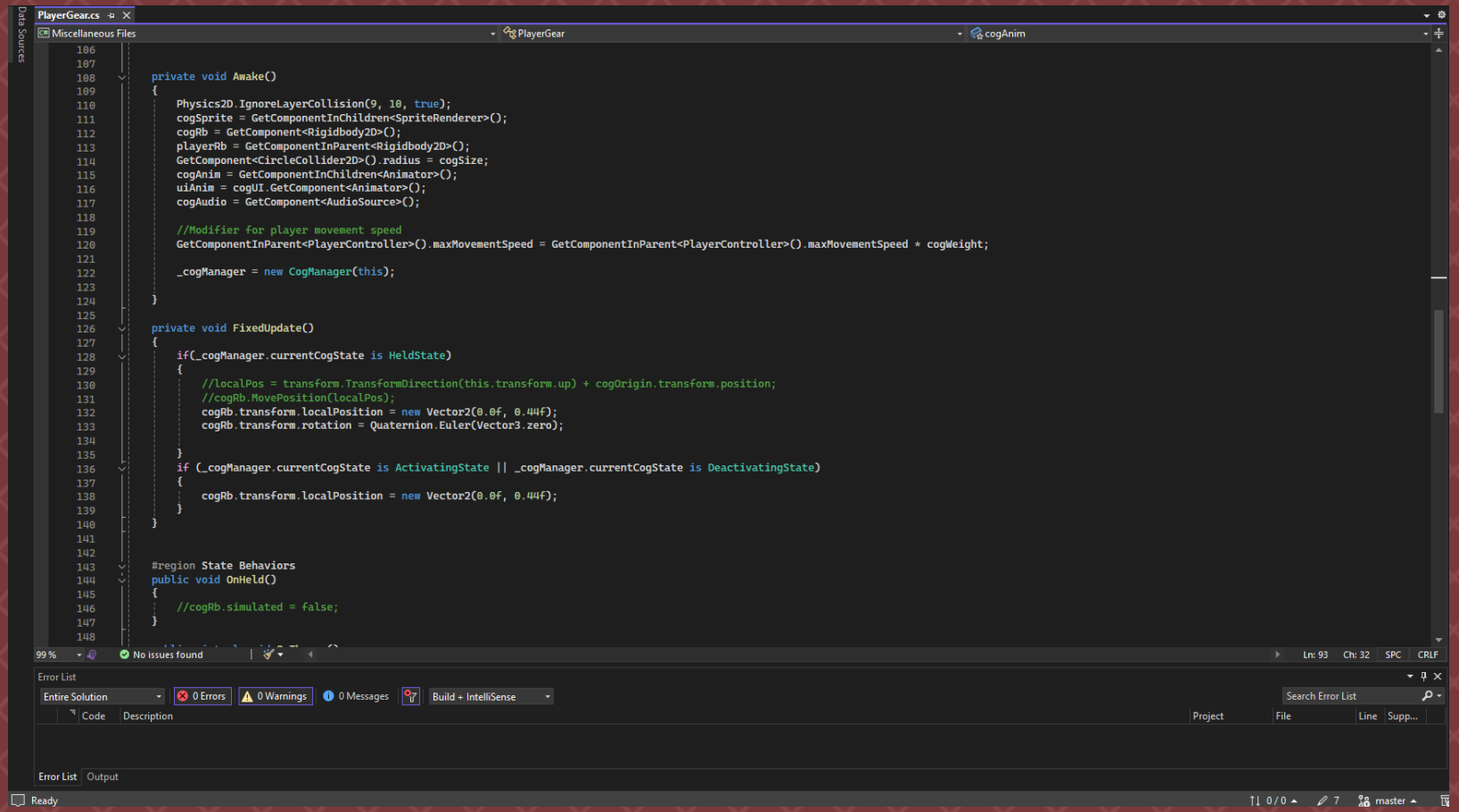
Prototyping examples



Visual scripting



Text-based scripting/programming


PlayerGear.cs
Miscellaneous Files
PlayerGear
cogAnim
private void Awake()
{
Physics2D.IgnoreLayerCollision(9, 10, true);
cogSprite = GetComponentInChildren<SpriteRenderer>();
cogRb = GetComponent<Rigidbody2D>();
playerRb = GetComponentInParent<Rigidbody2D>();
GetComponent<CircleCollider2D>().radius = cogSize;
cogAnim = GetComponentInChildren<Animator>();
uiAnim = cogUI.GetComponent<Animator>();
cogAudio = GetComponent<AudioSource>();
//Modifier for player movement speed
GetComponentInParent<PlayerController>().maxMovementSpeed = GetComponentInParent<PlayerController>().maxMovementSpeed * cogWeight;
_cogManager = new CogManager(this);
}
private void FixedUpdate()
{
if(_cogManager.currentCogState is HeldState)
{
//localPos = transform.TransformDirection(this.transform.up) + cogOrigin.transform.position;
//cogRb.MovePosition(localPos);
cogRb.transform.localPosition = new Vector2(0.0f, 0.44f);
cogRb.transform.rotation = Quaternion.Euler(Vector3.zero);
}
if (_cogManager.currentCogState is ActivatingState || _cogManager.currentCogState is DeactivatingState)
{
cogRb.transform.localPosition = new Vector2(0.0f, 0.44f);
}
}
#region State Behaviors
public void OnHeld()
{
//cogRb.simulated = false;
}
99 %
No issues found
Ln: 93
Ch: 32
SPC
CRLF
Error List
Entire Solution
0 Errors
0 Warnings
0 Messages
Build + IntelliSense
Search Error List
Code
Description
Project
File
Line
Error List
Output
Ready
master

Constants & Variables

Constants & Variables

- Integer (int) – Whole number
- Float (float) – Decimal number
- Character (char) – Single typed character (a, b, 1, 3, #, ?)
- String (string) – Collection of characters (abcd, 1dj?0--)
- Boolean (bool) – Value of either TRUE or FALSE

Engines may have data types unique to each of them!

References

Data structures

- Structure (struct) – General named collection of data types
- Class (class) – Means to contain data types and functions into a single location.
- Vector (Vector3) – Three floats, often representing position, rotation or scale.
- Transform (Transform) – Three vectors, often representing position, rotation and scale together.
- Array (type[size]) – A numbered collection of a single data type, starting from 0 upwards.

Some engines may not natively support some data structures.

Operators

- Addition (+) – Add two data types together
- Subtraction (-) – Subtract, as seen in math
- Multiply (*) – Multiply the first value by the second
- Increment (++) – Increase value of integer by 1.
- Decrement (--) – Decrease value of integer by 1.

Statements

- True/false – if(condition statement){}
- For – for(initial statement; condition statement; repeated operation){}
- For Each – foreach (type typeName in arrayName){}
- While – while(condition statement){}

Functions

```
private int RunTheMath(int a, int b){
        int value = a + b;
        return value;
}

public int main(){
        int result = RunTheMath(1,4);
        return 0;
}
```

Other things that might help

```
class Vehicle  // base class (parent)
{
  public string brand = "Ford";  // Vehicle field
  public void honk()             // Vehicle method
  {
    Console.WriteLine("Tuut, tuut!");
  }
}

class Car : Vehicle  // derived class (child)
{
  public string modelName = "Mustang";  // Car field
}

class Program
{
  static void Main(string[] args)
  {
    // Create a myCar object
    Car myCar = new Car();

    // Call the honk() method (From the Vehicle class) on the myCar object
    myCar.honk();

    // Display the value of the brand field (from the Vehicle class) and the value of the modelName
    Console.WriteLine(myCar.brand + " " + myCar.modelName);
  }
}
```

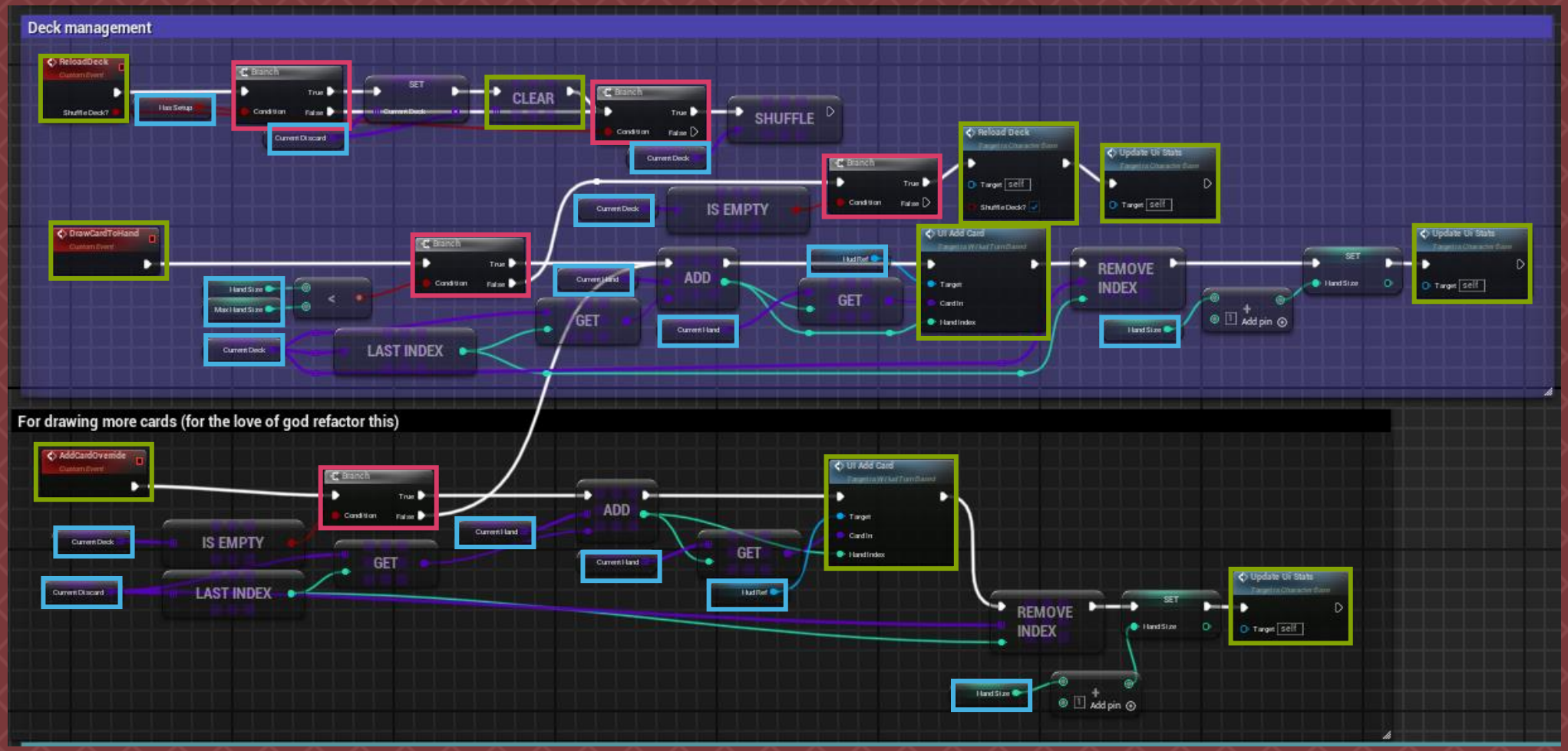
Inheritance

C#

```
interface ISampleInterface
{
    void SampleMethod();
}

class ImplementationClass : ISampleInterface
{
    // Explicit interface member implementation:
    void ISampleInterface.SampleMethod()
    {
        // Method implementation.
    }

    static void Main()
    {
        // Declare an interface instance.
        ISampleInterface obj = new ImplementationClass();

        // Call the member.
        obj.SampleMethod();
    }
}
```

Interfaces

An example


Deck management
ReloadDeck
Branch
SET
CLEAR
SHUFFLE
IS EMPTY
Reload Deck
Update UI State
DrawCardToHand
ADD
GET
LAST INDEX
UI Add Card
REMOVE INDEX
Add pin
For drawing more cards (for the love of god refactor this)
AddCardOverride

An example

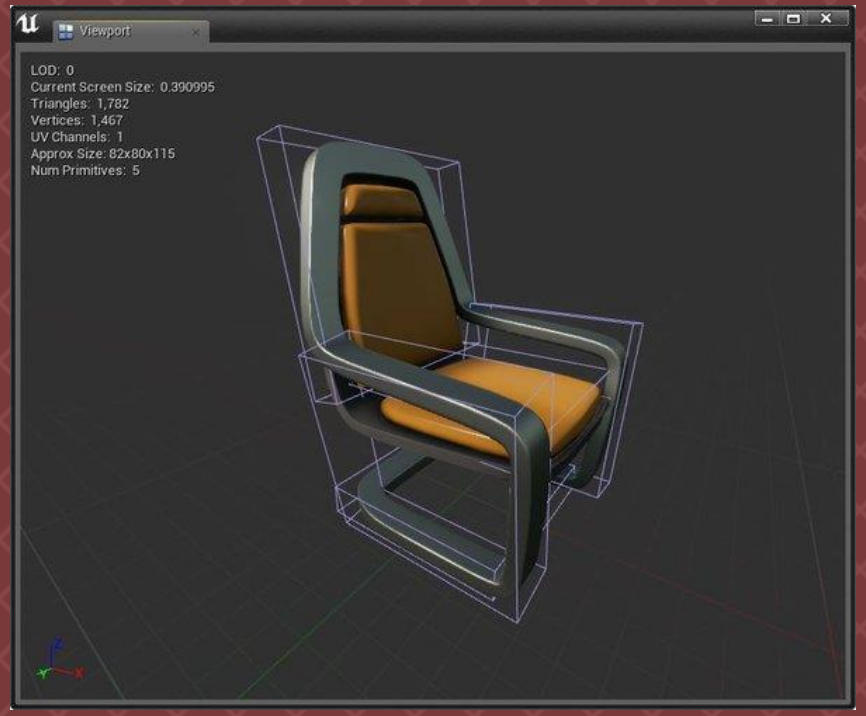

PLAYER TURN
6 HEALTH
30 COST
CHOOSE AN ACTION
EFFECT (-EV)
MOVE (+MV)
DISCARD (+EV)
BOOTS
MOVE +15
PUNCH
SNOOZE
COMMANDS
MOVE
END TURN
DECK
USED
7
0

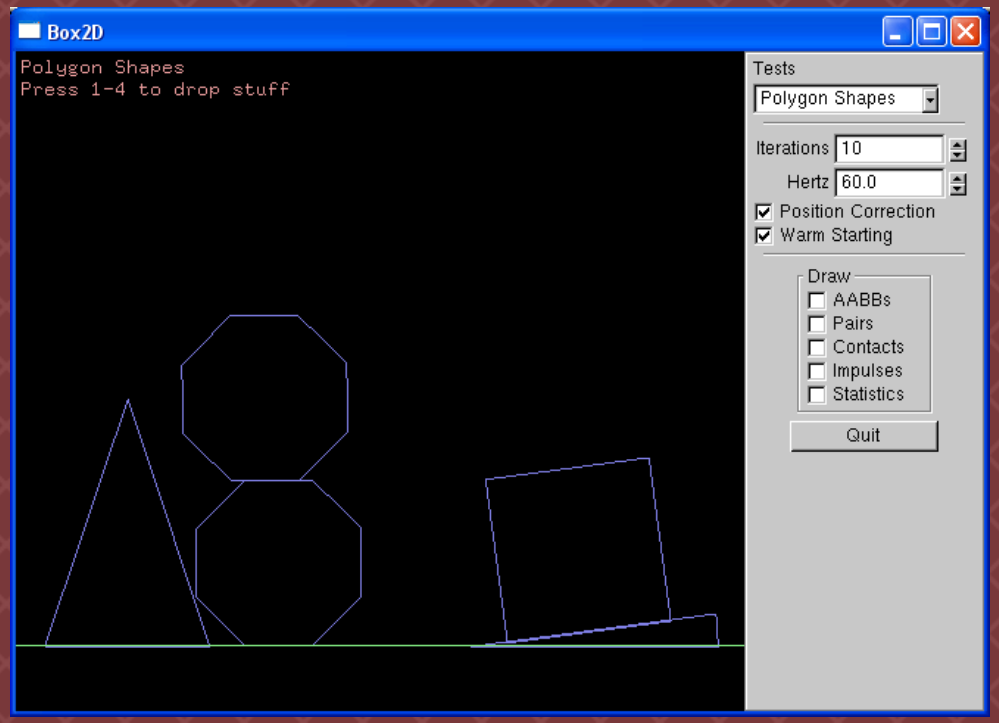
Optimization and how far is optimizing helpful?

KISS Principle

Keep
It
Simple,
Silly

Collisions


Viewport
LOD: 0
Current Screen Size: 0.390995
Triangles: 1,782
Vertices: 1,467
UV Channels: 1
Approx Size: 82x80x115
Num Primitives: 5


Box2D
Polygon Shapes
Press 1-4 to drop stuff
Tests
Polygon Shapes
Iterations 10
Hertz 60.0
Position Correction
Warm Starting
Draw
AABBs
Pairs
Contacts
Impulses
Statistics
Quit

Learn to Google!

- Prototyping is great for saving time during development.
- Visual scripting languages are specific to one engine.
- Text scripting languages are usable across engines.
- Games are made up of variables, functions and classes.
- There are countless functions available already in engines, you don't need to script absolutely everything.
- Scripting doesn't need to be complex!
- Only optimize when necessary.
- Collision volumes/bounds are everywhere in games.
- Physics engines will make development less scary.
- Google and forums are your best friends when documentation isn't.
- Expect to jump between design and development!
- Easy to learn, difficult to master.

Resources for game development

- Infallible Code YouTube Channel https://www.youtube.com/@InfallibleCode/videos
- W3Schools – C# Documentation https://www.w3schools.com/cs/index.php
- W3Schools – C# Syntax https://www.w3schools.com/cs/cs_syntax.php
- Unity – Download https://docs.unity3d.com/560/Documentation/Manual/InstallingUnity.html
- Unreal Engine 5 – Download https://www.unrealengine.com/en-US
- Godot – Download https://godotengine.org/download/windows/
- Gamemaker Studio – Download https://gamemaker.io/en/download
- Visual Studio 2022 (IDE) – Download https://visualstudio.microsoft.com/vs/
- Medium – Exploring functions and methods in C# https://medium.com/@praveen.rao.g.1990/exploring-functions-and-methods-in-c-8cbe5c2c3481
- Microsoft Learn – Interface https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/interface
- Unity Documentation – Physics https://docs.unity3d.com/Manual/PhysicsSection.html
- Stack Overflow – Programming/scripting forum https://stackoverflow.com

011 PART D Asset Creation

Creating art, animation and music for games.

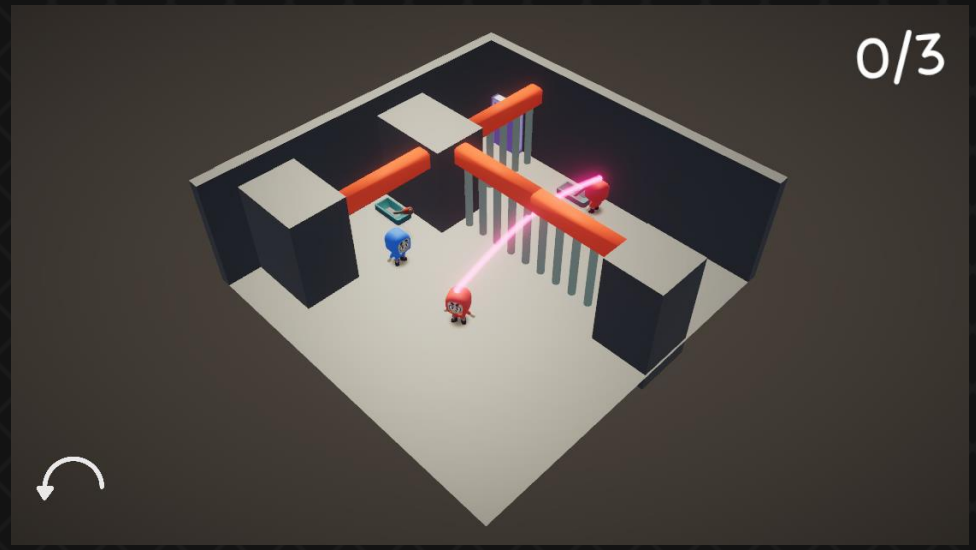
I’m not an artist!
OR
I have no interest in art…

Art is a complex skill, but not needed for games

- 'Programmer art'

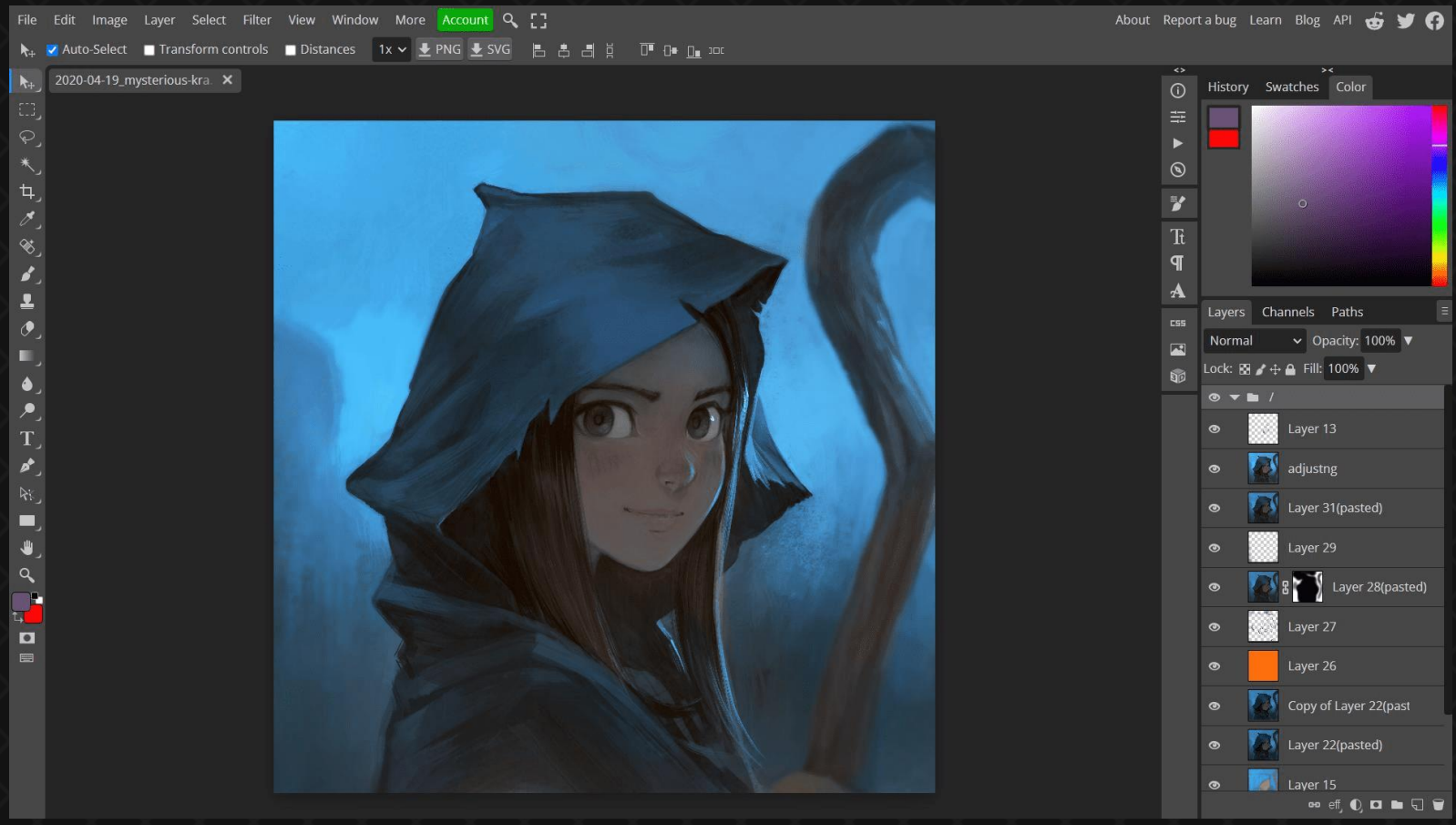
So, you want to make a game.

'Programmer art'



So, you want to make a game.

2D art

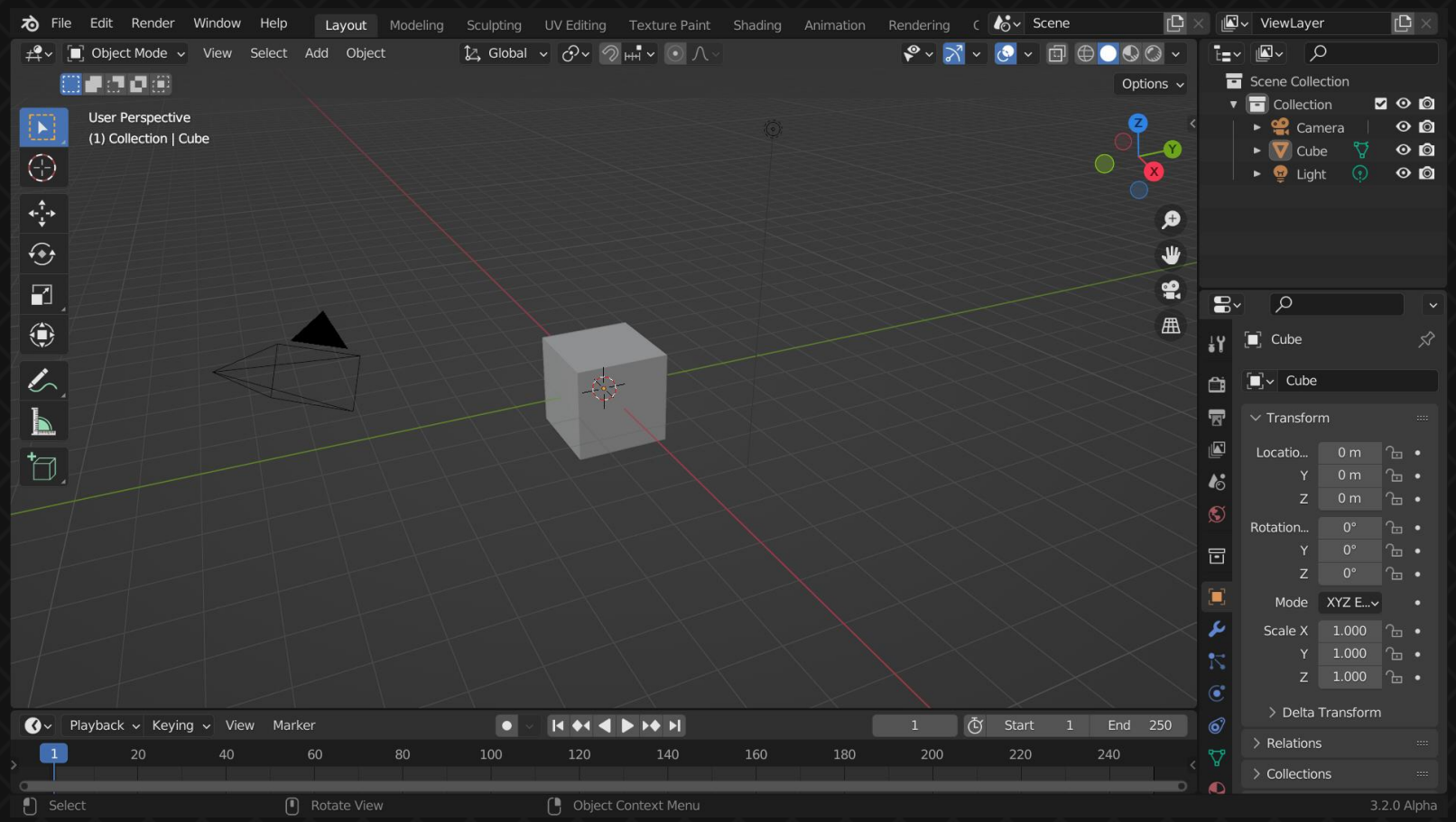

File Edit Image Layer Select Filter View Window More Account
About Report a bug Learn Blog API
Auto-Select Transform controls Distances 1x PNG SVG
2020-04-19_mysterious-kra
History Swatches Color
Layers Channels Paths
Normal Opacity: 100%
Lock: Fill: 100%
Layer 13
adjustng
Layer 31(pasted)
Layer 29
Layer 28(pasted)
Layer 27
Layer 26
Copy of Layer 22(past
Layer 22(pasted)
Layer 15


Base Color Map
Metallic Map
Normal Map
Roughness Map

Materials & shaders

So, you want to make a game.

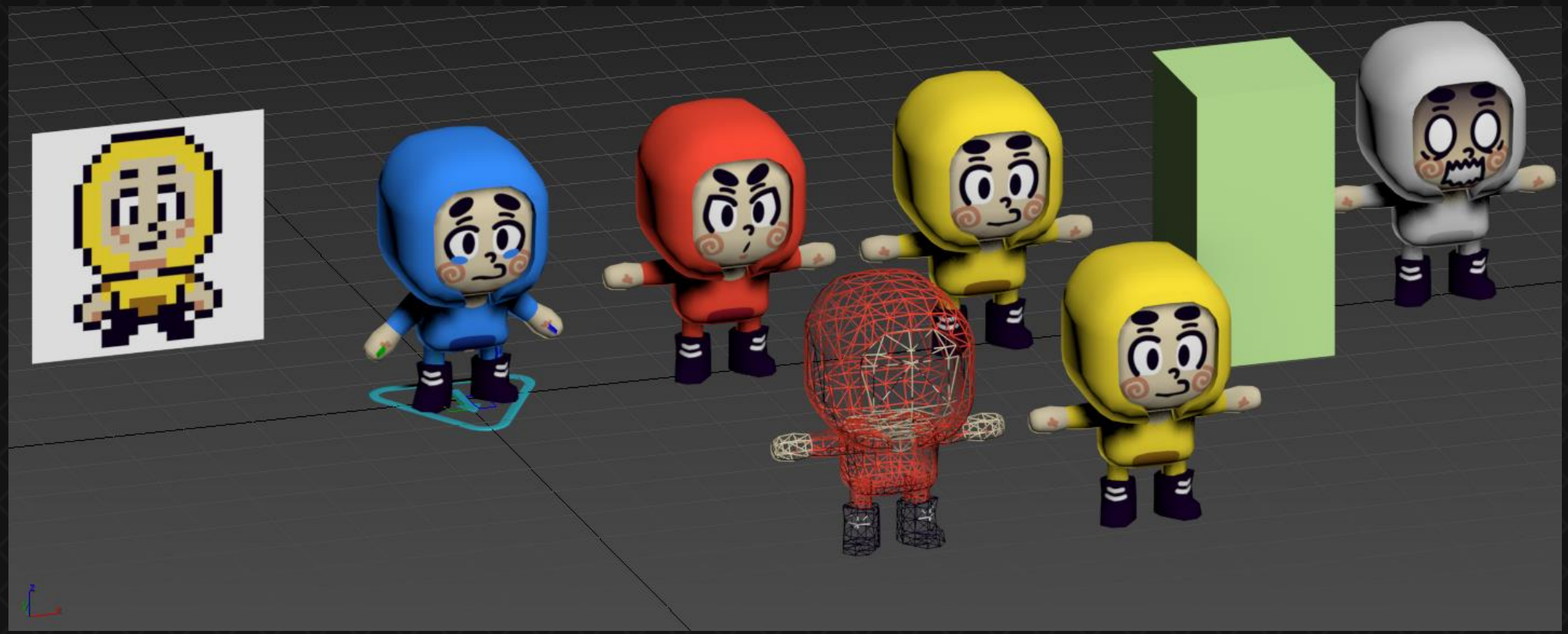
Modeling software



So, you want to make a game.

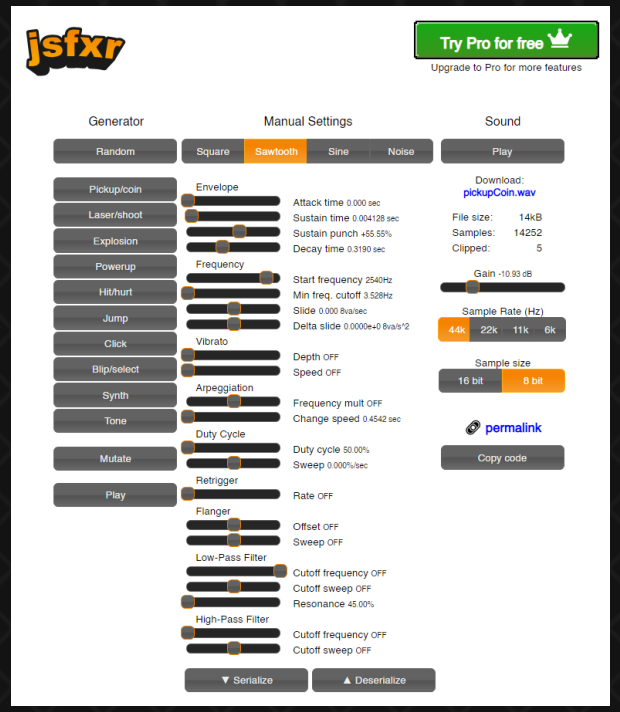
Modeling software

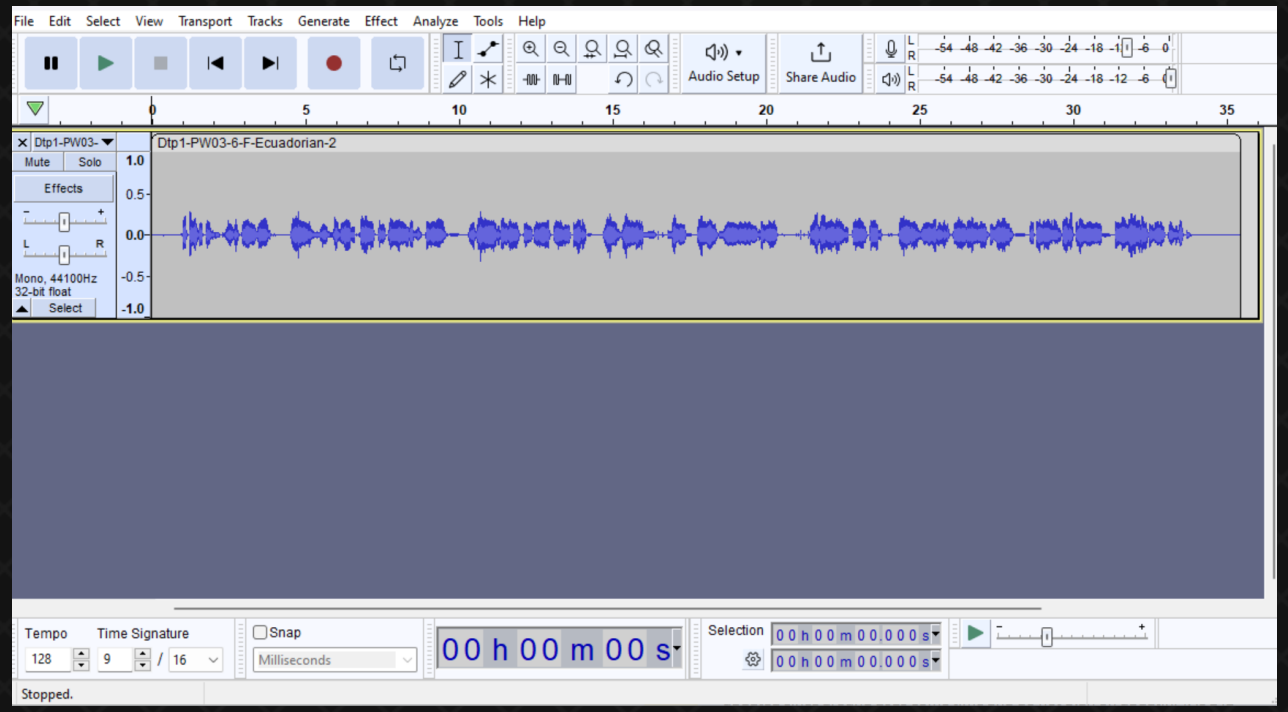
Fellow Bubs



So, you want to make a game.

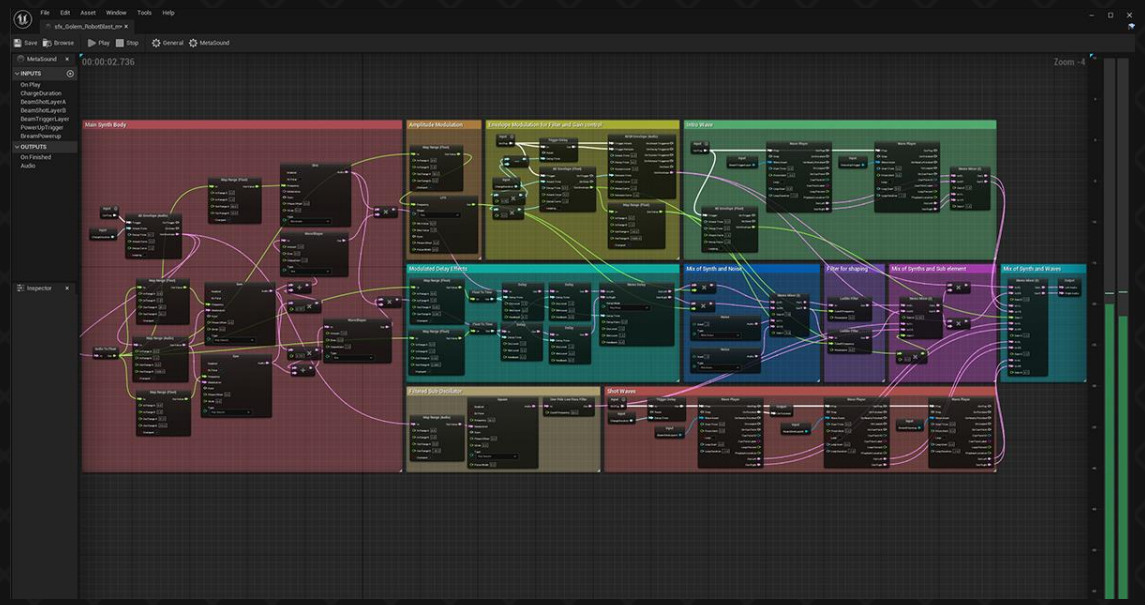
Anything else?

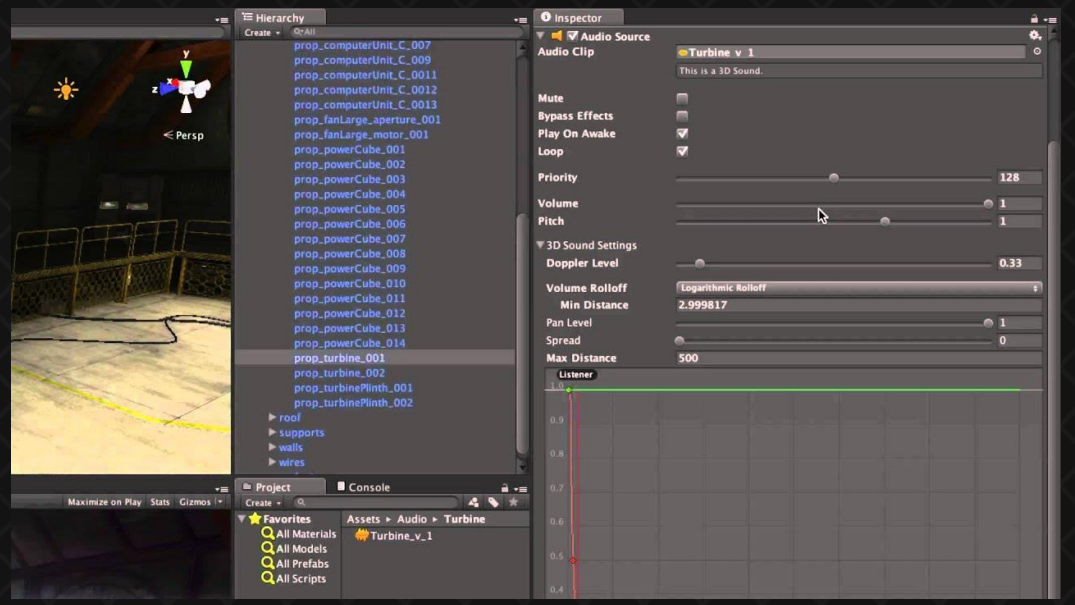




So, you want to make a game.

Sound in game engines



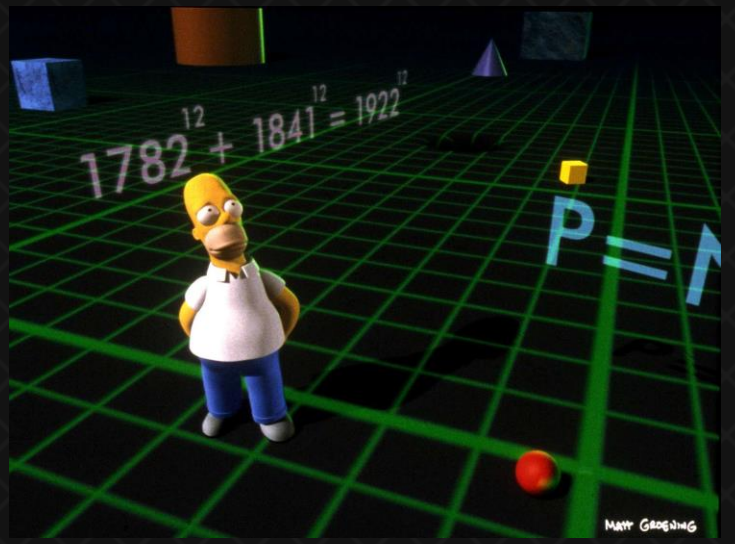


So, you want to make a game.

Sound optimization

- Art is a complex skill to master, but anybody can do it.
- Art will not make a bad game any better.
- There is plenty of amazing free-use software out there.
- Shaders will be a deciding factor in how your game looks.
- Everything is made from vertices!
- Work from concepts, save yourself.
- Making game sounds also doesn't have to be complicated.
- Asset optimization is important to a small game file size.

There are many other aspects to asset creation too, such as particle effects, simulations, lighting and so on…



Resources for asset creation

- Photopea https://www.photopea.com
- Blender – Download https://www.blender.org/download/
- Royal Skies YouTube Channel https://www.youtube.com/@TheRoyalSkies
- jsfxr – 8 bit sound maker https://sfxr.me
- Freesound – https://freesound.org
- Audacity – Download https://www.audacityteam.org/download/
- Acerola – PBR and Stylized Graphics https://www.youtube.com/watch?v=KkOkx0FiHDA

In conclusion...

Any questions?