# **[System Miami]**

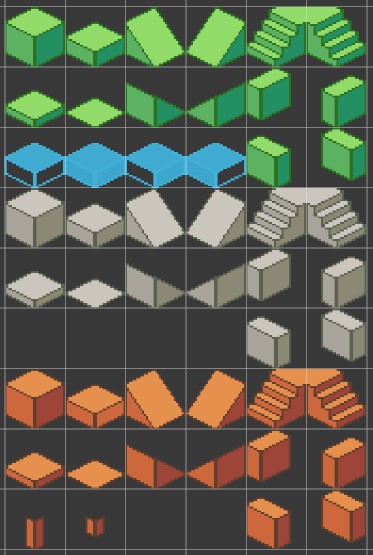
# **Production Bible**

## 

## **Art**

**General**

* 2D
* Pixel Art
* Isometric
* *Everything in the environment should be made of isometric tiles*
* This means no actual polygons or triangles.
* 2:1 Ratio
  + Tiles should fit into a sliceable square with dimensions equal to the larger side.
    - e.g. A 64 x 32 ‘pixel’ tile should exist inside a hypothetical square exactly 64 x 64.
    - Should be placed within the sprite sheets in a way that will allow stacking / combining, either manually or programmatically by generator scripts.



* + This 2:1 ratio is easiest to conceptualize with the (x, y) plane of a cube.
  + Example | Isometric Cube | (Top left sprite above)
    - Isometric Dimensions
      * X size (width): 1 “unit”
      * Y size (depth): 1 “unit”
      * Z size: (height): 1 “unit”
    - 64 x 64 Sprite
    - Consists of a top surface and 2 side surfaces
    - The top surface:
      * Exists on the (x, y) isometric plane, with dimensions (1, 1).
      * But in *screen space*, its dimensions are:
        + X (width): 64 “pixels” || 1 sprite-cell wide
        + Y (height): 32 “pixels” || 0.5 sprite-cells tall
* Objects can be larger than single squares, but their placement should be kept in mind so that they may be stacked onto the sides or tops of other “blocks.”
* ***\*\*Layla would like to recommend that artists and programmers alike read this article about isometric projection in video games\*\****
  + <https://pikuma.com/blog/isometric-projection-in-games>
  + Isometric projection is traditionally about making each of the three angles at the corner of the hypothetical cube 120°, but the angles are different in video game development *for the sole purpose of using a 2:1 pixel ratio*.

**Environment**

* Environment will consist of tiles placed procedurally by scripts. This means the work for artists will be in creating tilesets.
* Each “Area” will have its own art style, based on an area in Miami.
* Each Area will need the same tilesets, with the same shapes, fulfilling the same functions, (with a few possible unforeseen exceptions), but with their own color / flavor.
* Essentially each Area will have its own “skin,” differing only cosmetically Area to Area.
* Determining tilesets
  + Artists can determine what goes into these tilesets by creating buildings first, then deciding how to divide them into tiles.
  + Or they can use the guidelines below.
  + Either way, artists should feel free to create more decor or tile types as they see fit
    - To mesh with their general visual ideas
    - As long as the sprites will fit nicely into an isometric tile system.
  + Devs will find ways to implement artists’ rules if they communicate them. Examples:
    - “Place a banner / awning every 5 tiles on this type of building.”
    - “This building should only be 10 blocks wide.”
    - “Only put windows on the street-facing side of this type of building.”
    - “This building type should only have odd-numbered dimensions.”
    - “Only put this type of block directly over the entrance.”

**Tileset Guidelines**

* Bidirectional Blocks
  + Different shading on each side (Left/Right/Top).
  + Shading can be as simple as turning down RGB values a bit.
  + Unused sides will be covered up by the next block.
* Decor (Non-block Sprites)
  + One left-facing, one right-facing.
  + Sprites can be mirrored, which means:
    - If desired, artists can choose to make shading the only difference between the left-facing and right-facing variants.
    - Artists can choose to differentiate left-facing and right-facing in additional ways beyond shading if they see reason to.
* **Each Area Has**:
  + Buildings
    - 3 - 5 Building Type Tilesets
      * (e.g. Skyscraper, Metrorail Station, parking garage, residential, coffeeshop, etc.)
      * Each Building Type Tileset has:
        + Ground Floor Block
        + Normal Block
        + Corner Block (optional)
        + Small Entrance Block
        + Large Entrance Block
        + Roof Block
        + Unique Block

(e.g Support Beam, Balcony, etc.)

Optional, as many as desired

* + - 1 Window Tileset
      * (e.g. Small Window, Medium Window, Big Window, etc.)
      * 2 - 4 Window Types
        + Each with left-facing sprite, right-facing sprite.
      * Can be placed on Normal Blocks of any Building Type
    - 1 Entrance Overlay tileset
      * A Decor Object
        + (e.g. Screen? Sigil? Rune?)
        + Left-facing sprite, right-facing sprite
        + To be placed on a Building, near the entrance.
        + Something with 3 variants which signify differing levels of difficulty.

We’re thinking green, yellow, red to represent the difficulty levels, but any creative way to differentiate them would be great.

* + - * Can be placed on Normal Blocks of any Building Type
      * Each Entrance Overlay Tileset includes:
        + Easy Variant
        + Medium Variant
        + Hard Variant
    - Building Decor Tileset
      * Miscellaneous
      * Area-dependent
      * Can be placed on Normal Blocks of any Building Type
      * Can either be placed anywhere or dependent on building type, if the artists prefer.
      * 5 - 10 Building Decor Objects
        + Each with left-facing sprite, right-facing sprite
      * Example Building Decor Tileset:
        + Ivy / vines
        + Awning 1
        + Awning 2
        + Gargoyle
        + Banner 1
        + Banner 2
        + Trim / flourish
        + Whatever you can think of that fits the neighborhood!
  + Ground
    - 1 Ground Block Tileset
      * (e.g Cement, Dirt, Grass, Pavement, Brick, etc.)
      * These should be blocks at quarter height, rather than full cubes.
      * Ground can either be dependent on building type, if the artists prefer, or composed of mostly homogenous regions based on other factors.
      * Ground Block Tileset has:
        + Main Ground Block
        + Alternate Ground Block 1
        + Alternate Ground Block 2
        + Alternate Ground Block 3
        + Alternate Ground Block 4
    - 1 Sidewalk Block Tileset
      * These should be blocks at quarter height, rather than full cubes.
      * Sidewalk Block Tileset has:
        + Main Sidewalk Block
        + Sidewalk Curb 1 (upper right curb)
        + Sidewalk Curb 2 (upper left curb)
        + Sidewalk Curb 3 (lower left curb)
        + Sidewalk Curb 4 (lower right curb)
    - Ground Decor Tileset
      * Miscellaneous
      * Area-dependent
      * Can be placed on Normal Blocks of any Building Type
      * ~10 Building Decor Objects
        + Each with left-facing sprite, right-facing sprite
      * Example Ground Decor Tileset:
        + Streetlamp
        + Dumpster
        + Tree 1
        + Tree 2
        + Abandoned car
        + Barricade
        + Fence
        + Fence corner
        + Litter 1
        + Rubble
        + Crack (can be overlaid on any ground tile)
        + Crosswalk
  + Roads
    - 1 Road Block Tileset
      * These should be blocks at quarter height, rather than full-size cubes.
      * Road Block Tileset has:
        + Normal Road Block
        + Alternate Road Block
    - 2 Road Overlay Tilesets
      * These should be transparent overlays that can be slapped on a Road Block
      * One White, one Yellow
      * Each Road Overlay Tileset has:
        + Single Solid Line Tile
        + Single Dashed Line Tile
        + Double Solid Line Tile
        + Double Dashed Line Tile
* **Combat Rooms:**
  + Each Building Type in an area should have its own Combat Room Tilesets
    - Colors correspond to the associated Building Type
    - Building Types which are visually / thematically similar enough can use the same tilesets.
  + Each Combat Room has:
    - 1 Game Board Block Tileset
      * These should be blocks at quarter height, rather than full cubes.
      * Game Board Block Tileset has:
        + Game Board Block A (lighter)
        + Game Board Block B (darker)
  + Tile Status Overlays
    - These are sprites that can be laid on top of a Game Board block when something happens to it.
    - Examples:
      * Cracked
      * Hole
      * Obstruction
        + (e.g. Rock, Rubble, Plant, etc.)
        + Themed with Building Type
    - This will probably mean brief animations for the onset, removal, or change of each of these conditions.

**Concept Art for Tile Condition Overlay Animations:**

* Created by Layla in Pixel Studio (avail for free on Windows and Mac)
* Using the same tile in the background, with everything else drawn on top.

|  |  |
| --- | --- |
| Game Board Tile (Normal → Cracked) | Game Board Tile (Cracked → Hole) |
|  |  |

## **Art Style References**

### **Buildings**



### The above image gives a general feel, but ultimately the style will be dependent on the Area.

* Proposed Areas (**Bold Areas** indicate bare minimum inclusions. Others are desired if the artists have time / inspiration)
  + **Downtown**
  + **Nightlife**

### **Character/NPC**

* Pixel Art
  + High-ish detail (i.e. NOT low-res / Mario Bros / 8-bit)
  + Isometric angles
* Animations (5-10 frames, 8 directions)
  + Idle
  + Walk-cycle
  + Run-cycle
  + Punch enemy
  + Swing a melee weapon at enemy
  + Throw an object
  + Shoot a bow & arrow
  + Cast a ranged spell
  + Cast a close spell
  + Cast a spell on self (e.g. for healing)
  + Use item
* Style
  + Modern streetwear, fantasy elements / weapons
  + Isometric angles
  + 8-Directional Movement

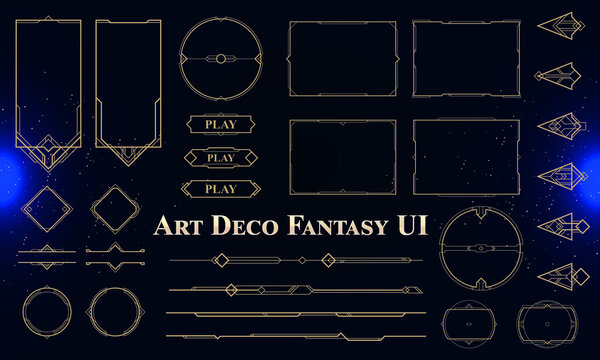
|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |

* Character creation
  + Very simple
  + 4 Clothing Sets (one per class, ~4 colorways each)
    - Fighter
      * T shirt
      * Overshirt? Rolled up sleeves?
      * Drawstring bag. One shoulder?
    - Tank
      * Heavy/puffy jacket or sweater
      * Cargo Khakis
      * Backpacking Rucksack
    - Mage
      * Loose
      * Flowy
      * Messenger bag / crossbody
    - Rogue
      * Lightweight
      * Hooded
      * Slim backpack
      * Darker colors (think Black Bloc style, but colors?)
  + ~5 hairstyles
  + ~5 hair colors
  + ~5 eye colors
  + ~5 skin colors

|  |  |
| --- | --- |
|  |  |
|  |  |

## 

## **UI**



### 

### **UI / UX Documentation**

### **Pause Screen**

### Character Viewer

### Walking animation

### Rotatable

### Menu Tabs

### Inventory

### Map

### Stats

### Settings

### Standard A/V settings options

### Accessibility (Color settings, subtitles, font, animation settings, etc.)

### **Combat**

### Turns

### Player turn or enemy turn

### Turn phase (movement, action)

### Player State

### Health, Stamina, Mana

### Attack, Magic, Item slots

### **Controls**

### Dynamic Panel with controls on screen at all times (**collapsible**)

### **Overworld**

### Constant Information On Screen:

### Health

### Stamina

### Mana

### Experience Level

### Experience Points

### Quickslot Inventory

### Variable Elements On Screen:

### Input / Interaction Prompts

### (?) and (!) to signify objectives.

### Symbols/Colors on Building doors signify difficulty level

### Green = Easy

### Yellow = Medium

### Red = Hard

### 

**Procedural Generation**

**General**

This style of programming will be challenging, but very rewarding. It will focus on creating scripts, prefabs, that perform the tasks of making decisions about what to place where, and will need to follow certain rules. It will require the creation of Class diagrams, flowcharts, and other data visualization and organization techniques.

**Systems**

* Area Generator
  + Programmers can set certain parameters like Intersection Size, Intersection Count, Building Density
  + Or programmers can set *ranges* for these parameters, and have the values generated at random within the ranges.
  + Hard-coded values or ranges can also be assigned for each Area, based on the feel of the neighborhood that we’re hoping to represent.
  + It will place intersections at certain origin coordinates throughout the game space.
* Intersection
  + Created and placed by Area Generator
  + Generates and manages:
    - Tile Region Generator
    - Ground
    - Roads
    - Buildings
  + Random number generation within certain parameters will inform the directions that it passes to Tile Region Generator.
  + Steps through an Enumerator to generate regions and objects.
  + Procedure

1. Ground-related
   1. Generate a square slab or ground the size of the intersection using the Tile Region Generator
   2. Generate any changes in the ground, or special ground areas. Using the Tile Region Generator
   3. Generate Roads | horizontal, vertical, or both
      1. Generate a horizontal road
      2. Generate a vertical road
2. Building-related
   1. Use a multitude of factors and RNG to determine origin points, dimensions, and types of buildings.
   2. Place Building seeds
      1. Step through the information generated in 2.a.
      2. Place a Building object at each predetermined origin point
      3. Initialize it with its dimensions and Type

* Tile Region Generator
  + Takes information such as Origin position, dimensions, and Region Type
  + Uses this information to create a Tile Region of a specific derived class, such as:
    - Homogenous Tile Region
    - Road
    - Building Floor
    - etc.
  + Steps through an Enumerator, creating a self-generating Tile Region.
  + Procedure
    - Create a Game Object
    - Add a specific component script derived from Tile Region based on the target Region Type
    - Initialize the Tile Region with its target origin and target dimensions.
    - Tell the region to start generating itself
    - Repeat
* Tile Region
  + Origin
    - Takes info from Tile Region Generator
    - Moves its transform to the given position
  + Dimensions
    - Passed from Tile Region Generator
  + Steps through an Enumerator to paint tiles on its tilemap, one by one
  + Procedure
    - For every Row
      * For every Column
        + Update the Target Tile Position to (Row, Column)
        + Determine which tile to place based on the target position.

This will be an abstract function that is overridden in each derived class.

This way, different Region Types can follow different rules, making their own determination of which tile to place.

* + - * + Place the determined tile at the target position.
* Building
  + Given origin and dimensions by the Intersection.
  + Uses a Tile Region Generator to construct a building layer by layer
  + Procedure
    - Create a footprint of the building
      * Create a new Game Object and give it a Tilemap
      * A layer of tiles that have no height (in isometric space)
      * Add a Tilemap Collider component ***to this layer only***.
    - For every floor (height dimension)
      * Create a new Game Object and give it its own Tilemap
      * This tilemap should be assigned an “Order in Layer” value equivalent to the iteration of this loop (i.e its “floor” number)
      * Pass origin, dimensions, and a Floor Type to Tile Region Generator
        + Somehow ensure a way that the script derived from Tile Region that gets added to the region corresponds to the right Building Type for the area, etc.
      * Reset the Tile Region Generator and repeat.
  + The lower the *screen space* Y coordinate of the building’s origin, the higher its priority in what it occludes.

The above systems and concepts should be modular enough to either use at face value, or replicate and tweak to do most of the work with Environment Generation for the whole game, including Combat Rooms. It’s a matter of knowing what information to feed them.

**Generation Rules**

* Areas
  + Generation depends on the player’s progression.
  + Number of encounters, and therefore intersections, increase as the player levels up and advances.
  + Shops spawn randomly between buildings (0 or 1 per Area).
  + Environment is generated within the scope of a theme for the Area.
* Buildings
  + Artists will create modular slices of buildings that can be rearranged to create a sense of randomness each time, minimizing the number of unique assets needed. These buildings will follow a theme based on their location in Miami, featuring recognizable landmarks to celebrate the city's history.
  + Basically, the artists will decide a lot of these rules as they get to creating their Tilesets.
* Combat Encounters
  + Environment / room generated at the time of commencing the interaction.
  + Determined in a range by current progression in the game:
    - Number of Enemies
    - Difficulty of Enemies
  + Rewards are also randomly generated (some are dependent on Character Class).
  + Tileset of Game Board dependent on
    - Current Area’s biome / theme
    - Artists decisions about these tilesets
  + Background Environment dependent on
    - Current Area’s biome / theme
    - Artists decisions about these tilesets

**Level Design**

* Areas are **procedurally generated** at runtime within set parameters.
* Grid/tile system.
* Each level will have procedurally generated dungeons that are either Easy, Medium, or Hard.
* Each dungeon will give a corresponding amount of Experience points

**Sparse animations**

* Mostly UI
* Rarely NPCs
* Fauna

### 

### **Tutorial (Level 0)**

* Based on Florida suburbs
* Simple two-street intersections that are limited and guided
* Two dungeon buildings and 3-4 duplicated buildings



*The tutorial will essentially be represented by the middle part of this picture.*

### **Level 1 (Downtown)**

* Sporadic car traffic to populate the area
* Buildings will be slightly dilapidated to signify the passage of time during the apocalypse and the breakdown of technology
* The Freedom Tower will be a permanent feature of the map



### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### **Level 2 (Wynwood)**



* One to two-story buildings decorated with street art of the artist's choice
* The landmark will be the Wynwood Walls

## **Monsters**

Monsters will have simple animations with only three actions: Idle, Move, and Attack. There will be five main monsters in the game: Slimes, Bats, Goblins, and Spiders. These will use the same sprite, except for Boss monsters which will be larger versions with slight variations. Artists will have creative freedom to design monsters.

### 

## **Intro**

* Character creation
* Combat tutorial
* Highlight UI elements

# **Mechanics**

## **Progression**

* Each Area has an Experience Level threshold that increments by five. (To move from Area 1 to Area 2, the player must have reached Experience Level 5)
* To level-up, a player must gain a certain amount of experience, obtained by completing combat encounters.
* When a player levels up, they are awarded two attribute points to upgrade any two attributes, which determine their stats (attack, defense, etc.).

## **Combat Mechanics**

Combat will be divided into phases, a movement phase and an action phase. The amount of spaces a character can move depends on their Dexterity.

* · Use Item – use an Item from Inventory that will grant a benefit.
* · Use Ability – use a special attack, defense,maneuver, or spell..
* · Attack – Basic attack, different animations depending on class but mechanically almost the same
* · **Defeat all enemies in the room to successfully complete the encounter.**
* · **Loss in combat rewinds progression back to entry to the Area (NOT the encounter).**
* **The first time they die there will be an on screen explanation of what happens.**

### **Items**

1. **Magical Whetstone**
   * **Description**: Increases weapon damage by a certain amount permanently

**2. Potion of Strength**

* + **Description**: Temporarily boosts the user’s strength, allowing them to deal more damage for 3 turns

3. **Potion of Speed**

* **Description**: A magical potion that speeds up the user, granting them an extra action for 2 turns

**Rewards**

* + EXP
  + Credits ($)
  + Items
  + Skill Points
  + Ultimate Medallion

## 

## **Class Descriptions/Specialties**

**Classes determine:**

* · Starting stats
* · Gear rewards from completion of encounters
* · What skills / spells one is allowed to acquire / buy
* · Character appearance throughout

Players can get abilities and spells as dungeon rewards.

### **Tank - Defense/Health**

**Abilities:**

1. **Shield Bash**
   * **Description**: The player charges forward a short distance, bashing an enemy with their shield, dealing moderate damage and stunning the enemy for 2 turns.
2. **Defensive Stance**
   * **Description**: The player adopts a defensive stance, reducing incoming damage by 50% for 2 turns..

### **Fighter - Damage**

**Abilities:**

1. **Power Strike**
   * **Description**: The player delivers a powerful strike with their weapon, dealing double damage to a single enemy.
2. **Whirlwind**
   * **Description**: The player spins in place, dealing damage to all enemies within a small radius.

### **Rogue - Stamina/DOT (Damage over Time)**

**Abilities:**

1. **Backstab**
   * **Description**: The player teleports behind an enemy and deals a critical hit, inflicting double damage.
2. **Poisoned Blade**
   * **Description**: The player coats their weapon with poison, causing their next attack to deal additional damage over time (DOT) for 3 turns.

### **Mage - Spells/Mana**

**Abilities:**

1. **Fireball**
   * **Description**: The player casts a fireball that explodes on impact, dealing damage to the target and nearby enemies.
2. **Magic Shield**
   * **Description**: The player summons a magical shield around themselves, absorbing a certain amount of damage for 2 turns.

**Stats System**

Stats determine how a player must approach combat, and will serve as an entry point for creating player interest.

There are 5 Attributes which determine the stats of the player in combat.

**STRENGTH**

* · Physical Attack / Skill Power
* · Skill Slots

**CONSTITUTION**

* · Health
* · Armor (Damage Mitigation)

**DEXTERITY**

* · Stamina (Resource for Skills)
* Determines speed(Number of spaces player is allowed to move per turn)

**INTELLIGENCE**

* · Mana (Resource for Spells)

**WISDOM**

* · Magic Attack / Spell Power
* · Spell Slots

### **Stats and Formulas**

#### **Strength**

* **Attack Power** (Base Effect 2.5):  
  (Strength \* 1.5) + Base Effect
* **Skill Slots** (1 Min, 6 Max):  
  If (Strength >= 5)  
  1 + (Strength - 5) / 4

#### **Wisdom**

* **Magic Power** (Base Effect 2.5):  
  (Wisdom \* 1.5) + Base Effect
* **Spell Slots** (1 Min, 6 Max):  
  If (Wisdom >= 5)  
  1 + (Wisdom - 5) / 4

#### **Dexterity**

* **Stamina**:  
  Dexterity \* 2

#### **Intelligence**

* **Mana**:  
  Intelligence \* 2

#### **Constitution**

* **Health**:  
  If (Constitution < 10)  
  Constitution \* 5  
  If (Constitution >= 10)  
  (Constitution \* 5) + 50
* **Defense (Damage reduction per hit)**:  
  Constitution