Full Documentation for PitSweeper

PitSweeper, a game released by a seemingly now-defunct *Kinsman Games*, was released in April of 2009 on various websites including *Kongregate* and *Newgrounds*. It was developed on *Adobe Flash* by Sean Givan, and served as a unique interpretation of *Microsoft*'s *Minesweeper*, combining its unique puzzle mechanics with typical ones discovered in the dungeon crawler genre.

This document serves as a collection of information about the design, implementation and known history of *PitSweeper*.

Written by Michael Warmbier

Dedicated to Eddie

Table of Contents

History

Not much is confirmed about the development of *PitSweeper*. From what can be found, PitSweeper was an original game developed sometime prior to its release in April 2009. Following its release and apparent then-popularity, a sequel was developed, as mentioned on the official website of Kinsman Games. Unfortunately, below is the only known image of this sequel.

PitSweeper received a relevantly significant amount of attention initially, gaining thousands of impressions on both *Newgrounds* and *Kongregate*, and seeing itself released on similar Flash game hosting platforms. Though, whether these releases were sanctioned or not, is unclear.

The original developer of the game has been contacted, but has yet to accept a request to interview.

Gameplay

The gameplay of *PitSweeper* consists of revealing hidden spaces, while also exploring the then-charted territory in search of **treasures** and **weapons** to deal with the various monsters that the player, referred to as **Sweeper**, will encounter throughout the game. The final goal is to defeat the Tyrant, located on the ninth level. The game relies on a turn-based system, with each action occurring allowing the game to advance.

The player is given four slots, two of which are for available **weapons** and two of which are for collectable **treasures**. The first weapon slot is reserved for melee weapons, while the second is for ranged. Each time an item is equipped or used, a turn is passed.

Additionally, the player is able to move in any adjacent direction that is not being hidden by a **cap** (an unrevealed tile). Revealing these caps will also pass a turn, as well as reveal **numbers** which indicate nearby danger:

Green Numbers: a treasure or wall is nearby.
Yellow Numbers: a mix of treasures, walls and enemies are nearby.
Red Numbers: at least one enemy is nearby.

These numbers act as hints to help the player navigate. The higher the number, the more elements are nearby, capping at a value of eight.

The primary goal of the player is to defeat the **Tyrant** on the final stage of the game. To do this, they are expected to collect items while defeating enemies and collecting **Gold** *[sic]* in order to gain **experience**.

Level Generation

Level generation relies on a combination of predesigned level map matrices and a bit of randomness. The randomness allows for different sets of walls to be placed, each set denoted by their own value. You can find each specific map's data matrix in the Internal Data section.

The method responsible for creating the board and each element on it is pretty straight forward. After using the predesigned level map as a reference for wall placement, the treasures, items (by their rarity), zombies and enemies are placed randomly throughout the board, increasing in their odds based on how many elements are already near their location. Before this, however, an `Exit`, the `Player` and a `GoldHoard` object are *always* spawned. If the level is the final one, level nine, then a `Tyrant` is also guaranteed to spawn.

The code below is a translation of the responsible method into C++, along with comments:

```
void randomLevel(TileBoard board, int levelNum) {
    Bitmap levelMap = null;
    int leveData[];
    int randomSelection[], wallTileValue;
    int randWidthVal, ranHeightVal;
    int playerStartX, playerStartY;

    this.tb = board;
    this.levelno = levelNum;

    goodcount = 0;
    badcount = 0;

    /* Use level number to retrieve bitmap */
    levelMap = new Bitmap(new bgarrays[param - 1](320, 320));
    board.backLayer.addChild(levelMap);

    /* Store random preset level arrays */
    levelData = levelArrays[Math.floor(Math.random() * levelarrays.length)];

    randomSelection = [1, 2, 4, 8. 16, 32, 64, 128];
    wallTileValue = randomSelection[Math.floor(Math.random * 8)];

    // Determine Wall Tiles (bitwise operation)
```

```
for(int i = 0; i < levelData; i++)
    if (Number(levelData[i]) & wallTileValue) {
        Wall * newWall = new Wall();
        board.addThingAtIndex(newWall, i);
    }

/* Determine Player Spawn */
bool goodSpawn = false;
while (!goodSpawn) {
    playerStartX = Math.floor(Math.random * board.across);  // Random value between width;
    playerStartY = Math.floor(Math.random * board.across); // Random value between height
    if (board.getThingAt(ranWidthVal, ranHeightVal) == NULL) { // If location is not taken
        board.addThingAt(Main,singleton.player, ranWidthVal, ranHeightVal); // Spawn Player
        goodSpawn = true;
    }
}
goodSpawn = false;
/* Determine Tyrant Spawn (level 9) */
if (levelNum == 9) {
    while (!goodSpawn) {
        randWidthVal = Math.floor(Math.random() * board.across);
        randHeightVal = Math.floor(Math.random() * board.down);
        /* Affirm exit spawns more than six tiles from the player */
        if (board.getThingAt(randWidthVal, randHeightVal) == null && board.DistanceBetween(ra
            param.addThingAt(new Exit(), randWidthVal, randHeightVal);
            goodSpawn = true;
        }
    }
}
goodSpawn = false;
/* Determine Tyrant Spawn (level 9) */
if (levelNum == 9) {
    while (!goodSpawn) {
        randWidthVal = Math.floor(Math.random() * board.across);
        randHeightVal = Math.floor(Math.random() * board.down);
        /* Affirm Tyrant spawns more than six tiles from the player */
        if (board.getThingAt(randWidthVal, randHeightVal) == null && board.DistanceBetween(ra
            param.addThingAt(new Tyrant(), randWidthVal, randHeightVal);
            goodSpawn = true;
        }
    }
}
goodSpawn = false;
/* Determine GoldHoard Spawn (level 9) */
if (levelNum == 9) {
    while (!goodSpawn) {
        randWidthVal = Math.floor(Math.random() * board.across);
        randHeightVal = Math.floor(Math.random() * board.down);
        if (board.getThingAt(randWidthVal, randHeightVal) == null) {
```

```
                param.addThingAt(new GoldHoard(), randWidthVal, randHeightVal);
                goodSpawn = true;
            }
        }
    }

    int tilesNearby = {
        [-1, -1],
        [-1,  0],
        [-1,  1]
        [0,  -1],
        [0,   1],
        [1,  -1],
        [1,   0],
        [1,   1]
    };
    Thing nearbyTile;
    float nearbyIntensity;
    bool nearbyInterest;

    /* Iterate through board */
    for (int i = 0; i < board.across; i++) {
        for (int j = 0; j < board.down; j++) {
            if (board.getThingAt(i, j) == NULL) { // Determine if tile is available
                /* Check nearby tiles */
                for (int k = 0; k < 8; k++) {
                    nearbyTile = Thing.getThingAt(i + tilesNearby[k][0], j + tilesNearby[k][1]);
                    // Increase chance of a new element spawning based on nearby elements
                    if (nearbyTile != NULL) {
                        if (nearbyTile == Wall) nearbyIntensity += .02;
                        if (nearbyTile == Enemy || nearbyTile == Treasure) nearbyInterest = true;
                        if (nearbyTile == Exit) nearbyIntensity += .15;
                    }
                }

                /* Random chance to add an enemy or treasure depending on total already placed */
                if (nearbyInterest) nearbyIntensity += 0.2;
                if (Math.random() < nearbyIntensity) addSomething(i, j);
            }
        }
    }

}
```

Code presented is written in C++. Original code is ActionScript.

Additionally, below is the method responsible for spawning everything else. Enemies are considered under `addEnemy()`, while everything else is under `addTreasure()`:

```
void addSomethint(int tileX, tileY) {
    badSituation = 0;
    goodSituation = 0;

    /* IF: situation is incredibly unfortunate */
    if (badcount > goodcount + 3) badSituation = 5;

    /* IF: situation is incredibly fortunate */
    if (goodcount > badcount + 3) goodSituation = 5;

    /* Roll 50% good and bad until situation leans to either side */
    while (badSituation < 5 && goodSituation < 5) {
        if (Math.random() <= .5) badSituation++;
        else goodSituation++;
    }

    /* IF: situation is just swell, spawn an enemy */
    if (goodSituation > badSituation) {
        addEnemy(tileX, tileY);
        badcount++;
    }

    /* IF: situation sucks, spawn treasure */
    if (badSituation > goodSituation) {
        addTreasure(tileX, tileY);
        goodcount++;
    }
}
```

Code presented is written in C++. Original code is ActionScript.

Because of this fairly complex level generation, a large amount of variety is offered. It is unlikely that the player will ever encounter the same circumstances. In terms of room variants, without including the items and enemies placed later, there are a total of **82** different types of levels a player may encounter.

Bestiary

The **bestiary** (referred to as such internally) is a the collection of enemies present throughout the game that the player may encounter. Each of them has different specifics and statistics associated with them:

Enemy	Passive Effect	HP	Strength	Agility	Armor	Description	Color Code
Ancestor	Only spawns from the ancestor sword.	12	0	1	0	This ancestor is angered that Sweeper abused the Ancestor Sword.	#AA7700
Bat	Movements are sporatic and random.	4	-2	2	0	Bats flutter about in dark places, in a mostly harmless manner.	#996633
Beast	N/A	28	3	3	1	A rampaging, lunging whirlwind of thick fur and sharp teeth and red, red eyes.	#660000
Behemoth	N/A	36	4	-2	2	A massive creature of scales, muscle, and sharp glaring eyes.	#00CC33
Cat	Able to walk on unrevealed tiles.	3	-2	4	0	How'd you get down here? Go away, kitty	#AA5511
Cultist	N/A	9	0	0	0	Cultists are crazed humans, sworn to protect and	#AA0000

Enemy	Passive Effect	HP	Strength	Agility	Armor	Description	Color Code
						serve the Tyrant.	
CultistLeader	N/A	15	2	2	0	This cultist has struggled to the top of the Tyrant's dire ladder, and gained the strength necessary to lead.	#AAAA00
Enlarged Badger	N/A	8	0	2	0	A badger is a fierce little animal - oh, wait. Maybe not that little.	#EEEEEE
Enlarged Rat	Has a 5% chjance on hit to apply poison.	5	-1	1	0	Rats have thrived down here, and grown large and strong.	#BB7700
Enlarged Serpent	Has a 25% chance on hit to apply poison.	11	1	2	0	These underground snakes have somehow grown to the size of humans.	#00CC00
Ghoul	Has a 25% chance on hit to apply stiff.	18	1	-1	0	Ghouls are lost humans, on the very threshold of	#33AA00

Enemy	Passive Effect	HP	Strength	Agility	Armor	Description	Color Code
						the undead world.	
Grunt	N/A	6	0	-1	0	Grunts are dull-witted monsters that trod towards the nearest enemy, then thump away.	#AA0000
Invisible Stalker	Completely invisible.	18	1	3	0	It's not as big an advantage as you'd think.	#000000
Iron Golem	May only attack every other turn.	25	5	-2	2	A giant mechanical stomper of a creature - slow, but unstoppable.	#DDDDDD
Mosquito	Has a 25% chance on hit to apply anemia.	1	-2	2	0	A tiny insect, mostly harmless; but sometimes it carries disease.	#DD0000
Reaper	N/A	999	7	7	0	The Reaper is a powerful, relentless spirit, who chases the dawdling and idling.	#555555

Enemy	Passive Effect	HP	Strength	Agility	Armor	Description	Color Code
Sentry	Cannot move.	10	2	0	1	A sentry doesn't move, but protects its space aggressively from any passers-by.	#EEEEEE
Skeleton	N/A	12	1	0	2	Being nothing but bones, a rattling old skeleton is its own armor.	#FFFFCC
Slime	Has a 15% chance on hit to discard player's weapon.	12	-1	-1	5	A liquid body shrugs off swords, and energy blasts - humble, but difficult to kill.	#00FF00
Spectre	Has a 25% chance on hit to remove player experience.	21	-1	3	0	Spectres are powerful ghosts, turned bitter and evil from loneliness.	#6666AA
Tyrant	N/A	50	7	5	0	THE TYRANT OF THE PIT ABSOLUTELY HAS TO DIE	#CC0000
Whirling Sword	May only attack	12	2	2	3	The whirling, animated	#3333FF

Enemy	Passive Effect	HP	Strength	Agility	Armor	Description	Color Code
	every other turn.					sword advances - slowly, but surely.	
Zombie	Has a 40% chance to fail to move.	4	-1	-3	0	Zombies are slow, shuffling, and weak.. but they are almost never alone	#338800

Note: The missing symbol for the "Invisible Stalker" is intentional, as per this enemies intended mechanic.

Each enemy is represented via text as a character. The specific character that represents them is their initial, with the capitalization of said initial reflecting their significance as a foe. Additionally, each enemy is given a **dice number** and **dice type**. These values are utilized to calculate their maximum hit through the following formula:

> Strength + DiceNumber • DiceType = MaxHit

This method of determining damage is used to convey a sense of randomness. The **dice type** represents how high of a number may be rolled, while the **dice number** is how many times a roll occurs, per attack. The final result is then added to the strength value to be considered as the effective damage for a turn. The difference of that value and the target's **defense** is then utilized to produce the final result, making the formula:

> (RolledDamage + Strength) - PlayerArmor = FinalDamage

Here, you can see the function as it is defined, translated from ActionScript to C++, with the irrelevant portions removed:

```
bool tryToHit(TileBoard w, int targetX, int targetY, Thing target) {
  auto chanceOfMiss, damage, armorCalc;

  chanceOfMiss = meleeChanceOfMiss(this, target);
```

```
    /* Attack Misses by Chance */
    if (floor(rand() % 100) < chanceOfMiss)  // play miss animation

    /* Attack Hits by Chance */
    else {
      damage = getAnimateDamage() - target.armor;
      if (damage < 0) armorCalc = 0;
      target.hp -= damage;

      if (target.hp <= 0) // play death animation
      sideEffect();
    }
  return true;
  }

  /* Note: Method never actually returns false. Meaning, a tryToHit() is always successful, regardl
```

Code presented is written in C++. Original code is ActionScript.

Within this formula, the method `getAnimateDamage()` is used to return the `(RolledDamage + Strength)` factor. It is defined as such:

```
  int getAnimateDamage() {
    int totalDamage = 0;

    for (int i = 0; i < this.dicenumber; i++)
      totalDamage += rand() % dicetype + 1;

    return totalDamage + this.str;
  }
```

Code presented is written in C++. Original code is ActionScript.

Wherein the previously mentioned randomness mechanic is defined. Additionally, the method `meleeChanceOfMiss()` is also utilized, leading into another relevant mechanic, **miss chance**.

Miss chance may be seen as the following function:

> IF: EnemyAgility > PlayerAgility THEN: Chance = 0%
> ELSE: Chance = 10 - (EnemyAgility - PlayerAgility) * 10
> IF: Chance > 50% THEN: Chance = 50%

Likewise, defined as C++ code:

```
void meleeChanceOfMiss(Thing, Thing) {
  auto var;

  if (Thing[0].agility > Thing[1].agility) var = 0;
  else var = 10 + (Thing[1].agility - Thing[0].Agility) * 10;

  if (var > 50) var = 50;
  return var;
}
```

Code presented is written in C++. Original code is ActionScript.

Lastly, all enemies determine their movement by looking for the direction the player is relative to them and moving one tile in their direction. If their path is blocked, either by a wall tile or a cap, they will try the same direction diagonally instead. The only exception is the "bat", which moves based on mostly randomness. Path finding is not implemented into their AI.

```
bool tryToMove(Tileboard board, int xPlayer, int yTarget) {
  Point selfPos, dir;
  bool moveSuccessful = false;
  int moveChoice[];

  /* If the target is outside of board, return false */
  if (xPlayer >= board.across || xPlayer >= board.down || yPlayer < 0 || yPlayer < 0) return fals

  /* Get Position of Self  */
  self = TileBoard.getIndexOfThing(this);

  /* Based on where the player is, store direction */
  if (self.x < xPlayer) dir.x = 1;
  if (self.x > xPlayer) dir.x = -1;
  if (self.y > yPlayer) dir.y = 1;
  if (self.y > yPlayer) dir.y = -1;
        // Iterate through motion table
        for (int i = 0; i < motiontable.length; i += 2) {

        // IF: motion table matches direction of player
        if (motiontable[i][0] == dir.x && motiontable[index][1] == dir.y) {

            // Sub-iterate tthrough sub-motion table <--- insanity
            for (int j = 0; j < motiontable[i + 1].length) {

                // Iterate through nearby tiles
                moveChoice = motiontable[i + 1][j];

                /* IF: Tile exists  */
```

```
                if (board.getTileAt(self.x + moveChoice[0], self.y + moveChoice[1]) != NULL)

                    /* IF: no Thing already occupies the Tile */
                    if (board.getThingAt(self.x + moveChoice[0], self.y + moveChoice[1]) == NULL

                        /* IF: Tile is within the board */
                        if(self.x + moveChoice[0] < board.across && self.x + moveChoice[0] >= 0 &

                            board.moveThingBy(this, moveChoice[0], moveChoice[1], false);
                            Main.singleton.dirtypoints.push(self);
                            Main.singleton.dirtyPoints.push(new Point(self.x + moveChoice[0], sel
                            moveSuccessful = true;
                            break;

                }

            }

        }

    }

    return moveSuccessful;
}
```

Code presented is written in C++. Original code is ActionScript.

The Player

Within the game's presentation the player character is referred to as *Sweeper*. While inherited from the same `Animate` object as `Enemy`, the `Player` object has several unique characteristics.

The player begins with one level and will gain a level every time they reach their experience goal or one thousand Gold. Each time the player earns a level, their experience goal will increase by one. When this happens, their max hitpoints will increase by three. If the parity of their new level is even, they will be given one extra agility stat as well, while an odd parity will increase their strength instead.

The player's default stats are all *zero*, with the only exception being their health. Their melee weapon slot is initialized with "Bare Hands" to start, while every other slot is initialized with `NoThing`.

Damage is calculated for the player towards enemies the same as it is for them. However, in a strange act of realism, some items do not consider the player's strength in this calculation; these items are both of the crossbows and the wands.

Landmarks

Landmarks are objects in the game the player may interact with; they are then given the option to pay Gold to receive an effect. If the player doesn't have enough Gold, they won't be able to use the landmark.

Landmark	Price	When Purchased	Description
Altar of Dice	75 Gold	Rolls two dice and pays the result multiplied by ten back to the player	Two dice rest on top of this altar. The presiding spirit is unknown.
Altar of Enduring	200 Gold	+1 HP	This altar is watched eternally by a patient and unmoving spirit.
Altar of Fleet	200 Gold	+1 Agility -1 Strength	This altar is watched by a spritely demigod of action and speed.
Altar of Might	200 Gold	+1 Strength -1 Agility	This altar is watched by a demigod of power and strength.
Vending Machine	50 - 100 Gold	Sells at random one of the following: Arrows Bolts Healing Potion Strength Potion Short Sword Light Armor	Today's Sale: <item> for only <price> Gold!

Affects

"**Affects**" *[sic]* are effects that may be applied to any given entity, such as the player or an enemy. They can be applied in various ways, including through items and/or enemy passives.

Affect	Passive Effect	Duration	Chance To End
Agility	+2 Agility	6 Turns	20%
Anemic	Prevents healing and strength boosts	6 Turns	20%
Drunk	+1 Armor -1Agility	6 Turns	15%

Affect	Passive Effect	Duration	Chance To End
Poison	-2 Strength	6 Turns	20%
Regen	+1 HP Per Turn	4 Turns	100%
Stiff	-2 Agility	6 Turns	20%
Strength	+3 Strength	6 Turns	20%

Oddly enough, each affect is designed to last at a minimum duration, but may continue with a 20% chance. This applies to both negative and positive affects, with the only exception being `AffectRegen`.

Itemary

The **itemary** *[sic]* (also referred to as such internally) is the collection of game elements categorized as "items".

Weapons are tools that may be found by the player to provide methods of damage. Each one has its own unique stats and potentially its own passive.

Weapon	Rarity	Passive Effect	Chance to be Modified?	Ranged?	Ammo Type	Effective Range	Max Range
Ancestor Sword	Rare	Chance to spawn an Ancestor.	Yes	No	N/A	1	1
Battle Axe	Uncommon	+1 Agility	Yes	No	N/A	1	1

Weapon	Rarity	Passive Effect	Chance to be Modified?	Ranged?	Ammo Type	Effective Range	Max Range
Broad Sword	Uncommon	N/A	Yes	No	N/A	1	1
Crossbow	Uncommon	N/A	Yes	Yes	Bolts	10	25
Fire Wand	Rare	Has between an 8% and 14% chance to burn out every turn.	Yes	Yes	N/A	10	25
Glass Sword	Rare	Has a 15% chance to break every turn.	Yes	No	N/A	1	1
Katana	Rare	Has a chance to deal x2 damage.	Yes	No	N/A	1	1

Weapon	Rarity	Passive Effect	Chance to be Modified?	Ranged?	Ammo Type	Effective Range	Max Range
Lightning Wand	Rare	Has between an 8% and 14% chance to burn out every turn.	Yes	Yes	N/A	10	25
Long Bow	Uncommon	N/A	Yes	Yes	Arrows	8	25
Partisan	Uncommon	+1 Agility	Yes	No	N/A	1	1
Short Bow	Common	N/A	Yes	Yes	Arrows	4	20
Short Sword	Common	N/A	Yes	No	N/A	1	1
Sparkle Sword	Rare	NA	Yes	No	N/A	1	1

Weapon	Rarity	Passive Effect	Chance to be Modified?	Ranged?	Ammo Type	Effective Range	Max Range
Stick	Common	N/A	No	No	N/A	1	1
Sword	Uncommon	N/A	Yes	No	N/A	1	1
Twin Crossbow	Rare	Uses twice the ammunition.	Yes	Yes	Bolts	10	25

Randomness is used to determine many factors, such as damage and a potential to break, depending on the specific item. Additionally, all items, save for the "stick", have a 10% chance to be given the "nice" modifier, as well as a 15% chance to be given the "poor" modifier. These modifies add and remove one from the weapon's max hit, respectively.

While not listed here, as it is not organized as a weapon, the `BareHands` object is also an application of the weapon classification, and is the default melee weapon given to the player. It's only notable trait is its max hit of two.

Additionally, some items are considered **treasures**. These include both consumables and non-consumables, which both must be equipped before the player may gain any benefit. Only two of these treasures may be equipped at a given time.

Treasure	Uncommon/Rare	Passive Effect	Consumable?	Duration	Uses	Description
Dexterity Tonic	Uncommon	+2 Agility	Yes	At Least 6 Turns	1	The oil-like tonic limbers up your muscles.
Ring of Agility	Uncommon	+1 Agility	No	∞ Turns	∞	This ring has a latent magic that reinforces the body.
Cap Breaker	Uncommon/Rare	Reveals walls.	Yes	Instant	1	This device will uncover all tiles containing walls on the level.
Healing Concentrate	Uncommon/Rare	+ 50 HP	Yes	Instant	1	This dark green healing potion is extremely powerful.
Healing Keg	Uncommon/Rare	+8 HP	Yes	Instant	2 - 6	The Healing Keg contains multiple drinks worth of healing potion.
Healing Potion	Common	+8 HP	Yes	Instant	1	Healing potions repair

Treasure	Uncommon/Rare	Passive Effect	Consumable?	Duration	Uses	Description
						wounds and restore health when drunk.
Heavy Armor	Uncommon	+3 Armor -2 Agility	No	∞ Turns	∞	Metal plates clank together, impeding your movement but protecting handily.
Light Armor	Common	+1 Armor -1 Agility	No	∞ Turns	∞	Leather straps and buckles offer mild protection to the user.
Milk	Common	Removes status effects.	Yes	Instant	1	Delicious, healthy milk washes out the blood and removes status effects.
Mithril Armor	Rare	+3 Armor -1 Agility	No	∞ Turns	∞	You can always make some nice stuff when you're

Treasure	Uncommon/Rare	Passive Effect	Consumable?	Duration	Uses	Description
						using mithril.
Panic Button	Uncommon/Rare	Randomly relocates player to visible tile.	Yes	Instant	1	This device will teleport Sweeper to a random location in the level.
Ring of Regeneration	Rare	+1 HP (Per Turn)	No	∞ Turns	∞	The holder of this ring slowly has their wounds repair over time.
Resealant	Uncommon/Rare	Surrounds player with wall tiles.	Yes	Instant	1	This device surrounds Sweeper with a ring of tile caps.
Ring of Resilience	Uncommon	+2 Max HP	No	∞ Turns	∞	This ring has latent magic that reinforces the body.
Strength Potion	Uncommon/Rare	+3 Strength	Yes	At Least 6 Turns	1	This tonic will set your muscles on fire! But in a good way.

Treasure	Uncommon/Rare	Passive Effect	Consumable?	Duration	Uses	Description
Ring of Strength	Uncommon	+1 Strength	No	∞ Turns	∞	This ring has latent magic that reinforces the body.
Whiskey	Uncommon	+1 Armor -1 Agility	Yes	Instant	1	You can't feel the pain. But you also can't feel your legs

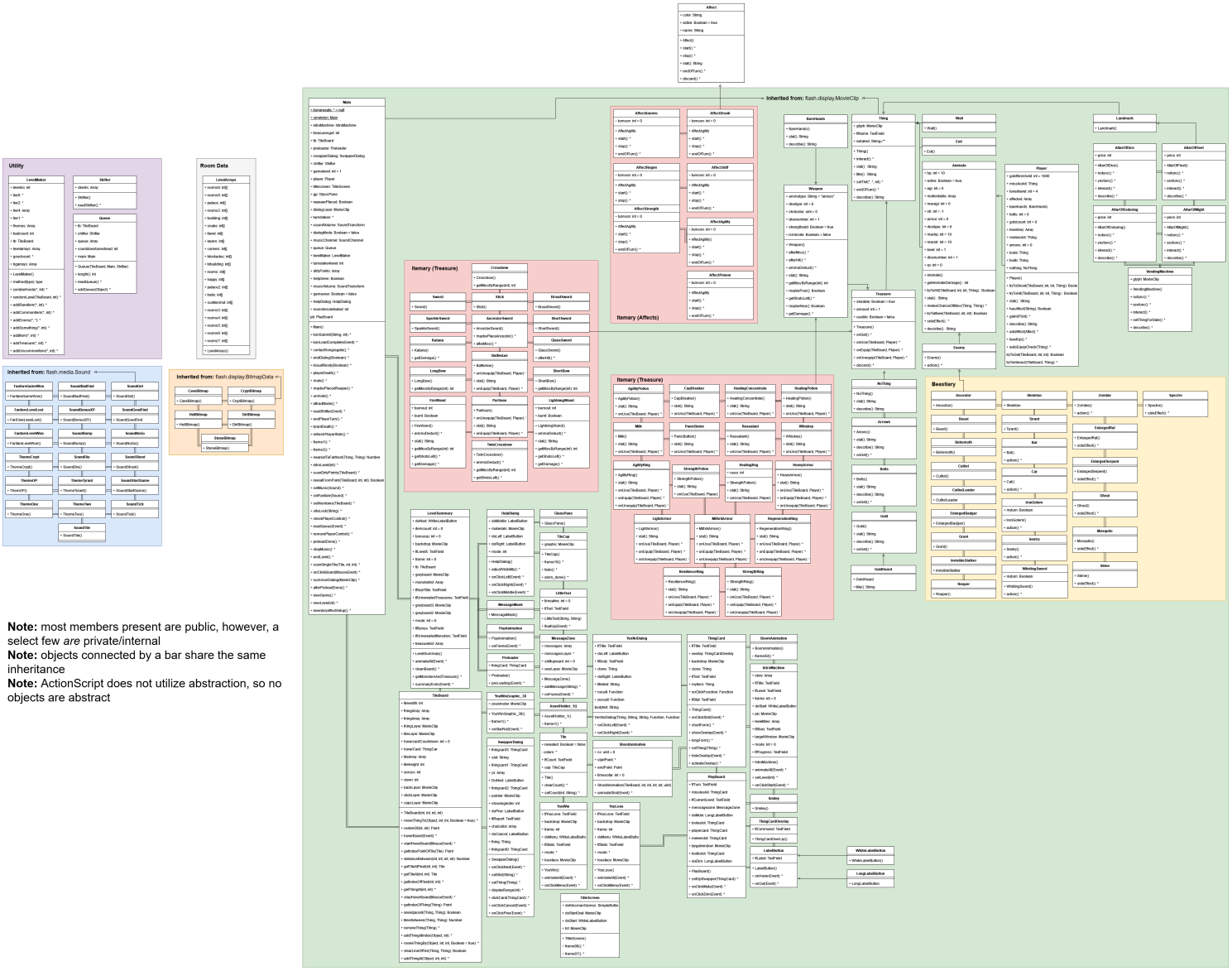
Similar to "Bare Hands", there are three treasures ("Gold", "Arrow" and "Bolt") which are excluded from this list aswell as the grouping of `Treasure` objects despite inheriting its class. The `Arrow` and `Bolt` objects are used as ammo for ranged weapons, while `Gold` (and its child, `GoldHoard`) is used statistically to increase the level of the player and as game score of sorts. Additionally, a treasure named `NoThing` is used as a default for both item slots, as well as the range slot.

Internal Data

This document and section covers the internal data stored within the game's scripts, it does not go over the sprite, image or text data found within the game.

Class Structure

The class structure within the game is rather straight forward; most classes inherit attributes from a master-class of sorts with implied abstractness (although ActionScript does not support this, a similar result is achieved via overriding empty functions). This class diagram in its entirety can be seen here:



Note: most members present are public, however, a select few *are* private/internal
Note: objects connected by a bar share the same inheritance
Note: ActionScript does not utilize abstraction, so no objects are abstract

Level Map Data

Each map in the game is randomly selected from a predesigned list. No map is hard coded to appear the same, so the user may encounter the same map on different levels, even within the same run. These levels are stored as arrays of data, which are intended to be read as matrices. Below are those matrices formatted to be clearer to understand. Keep in mind, the digit `0` represents a space where a wall may not appear, while any other number has a chance to represent one that will.

```
rooms8
0,   0,   0,   0, 255, 255, 255, 255,   0,   0,   0, 255, 255, 255, 255, 255,
0,  64,  64,   0, 255,   0,   0, 255,   0,   0,   0,   0,   0,   0,   0, 255,
0,  64,  64,   0, 255,   0,   0, 255,   0,   0,   0,   0, 255, 255, 255, 255,
0,   0,   0,   0, 255,   0,   0, 255,   0,   0,   0, 240, 240, 240, 240, 240,
0,  64,  64,   0,   0,   0,   0, 255,   0,   0,   0, 240,   0,   0,   0, 240,
0,  64,  64,   0,   0,   0,   0, 255,   0,   0,   0,   0,   0,   0,   0, 240,
0,   0,   0,   0,   0,   0,   0, 255,   0,   0,   0,   0,   0,   0,   0, 240,
```

```
 15,  15,  15,   0, 255,   0,   0, 255,   0,   0,   0,   0,   0,   0,   0, 240,
 15,   0,  15,   0, 255,   0,   0, 255,   0,   0,   0, 240,   0,   0,   0, 240,
 15,   0,   0,   0, 255, 255, 255, 255,   0,   0,   0, 240, 240, 240, 240, 240,
 15,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
 15,   0,   0,   0,   0,   0,   0,  12,  12,  12,  12,   0,   0,   0,   0,   0,
 15,   0,   0,   0,  15,  15,   0,  12,   0,   0,  12,   0,   0,   0,   0,   0,
 15,   0,   0,   0,   0,  15,   0,  12,   0,   0,   0,   0,   0,   0,   0,   0,
 15,   0,   0,   0,   0,  15,   0,  12,   0,   0,   0,   0,   0,  12,   0,   0,
 15,  15,  15,  15,  15,  15,   0,  12,  12,  12,  12,  12,  12,  12,   0,   0,
```

rooms9

```
255, 255, 255, 255,   0,   0,   0,   0,   0,   0,   0,   0, 255, 255, 255, 255,
255,   0,   0, 255, 255,   0,   0,   0,   0,   0,   0, 255, 255,   0,   0, 255,
255,   0,   0,   0, 255,   0,   0,   0,   0,   0,   0, 255,   0,   0,   0, 255,
  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
  0,   0,   0,   0,  15,   3,   3,   3,   3,   3,   3,  15,   0,   0,   0,   0,
255, 255, 255,   0,  12,   0,   0,   0,   0,   0,   0,  12,   0, 240, 240, 240,
255,   0, 255,   0,  60,  48,  63,  63,  63,  63,  48,  60,   0, 240,   0, 240,
255,   0,   0,   0,  12,   0,  15,   0,   0,  15,   0,  12,   0,   0,   0, 240,
255,   0,   0,   0,  12,   0,  15,   0,   0,  15,   0,  12,   0,   0,   0, 240,
255,   0,   0,   0, 204, 192, 207, 192, 192, 207, 192, 204,   0,   0,   0, 240,
255,   0, 255,   0,  12,   0,   0,   0,   0,   0,   0,  12,   0, 240,   0, 240,
255, 255, 255,   0,  15,   3,   3,   3,   3,   3,   3,  15,   0, 240, 240, 240,
  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
255,   0,   0,   0, 255,   0,   0,   0,   0,   0,   0, 255,   0,   0,   0, 255,
255,   0,   0, 255, 255,   0,   0,   0,   0,   0,   0, 255, 255,   0,   0, 255,
255, 255, 255, 255,   0,   0,   0,   0,   0,   0,   0,   0, 255, 255, 255, 255,
```

palace

```
255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
255,   0,   0,   0,   3,  12, 204,  12,  12,  12,  12,   3, 192,   0,   0, 255,
255,   0,  48,  48,  51,  60, 192,   0,   0,   0,  12,   0, 192,   0,   0, 255,
255,   0,  48,   0,   0,  12, 204, 192,   0,  12,  12, 195, 192,   0,   0, 255,
255,   3,  51,   3,   3,  48,   0,   0,   0,   0,   0,   3,   3,   3,   3, 255,
255,   0,  48,  48,  48,  48,   3,   3,  51,  51,  48,  48,  48,  48,  48, 255,
255, 192, 192, 192, 192,   0,   3,   0,  48,   3,   0,   0,   0,   0,   0, 255,
255,   3,   3,   3, 195,   0,   3,   0,  48,   3,   0,   3, 195, 195, 195, 255,
255,   0,   0,   0,   3,   0,  15,   0,   0,  15,   0,   3, 192,   0,   0, 255,
255,  12,  12,  12,   0,   0,  12,   0,   0,  12,   0,  12,  12,  12,  12, 255,
255,   0,   0,  12,   3,   0,  12,  12,  12,  12,   0,  15,   0,   0,   0, 255,
255,   3,   3,  15, 195,   0,   0,   0,  48,  48,  48,  63, 243,  51,  51, 255,
255,   0,   0,   0, 192,   0,   0,   0,   0,   0,   0,   0, 192,   0,   0, 255,
255, 255, 255, 255, 255, 255,   0,   0,   0,   0, 255, 255, 255, 255, 255, 255,
  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
```

rooms2

```
  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
  0, 255, 255, 255, 255, 255, 255, 255, 255,   0,   0,   3,   3,   3,   3,   3,
  0, 255,   0,   0,   0,   0,   0,   0, 255,   0,   0,   3,   0,   0,   0,   3,
  0, 255,   0,   0,   0,   0,   0,   0, 255,   0,   0,   0,   0,   0,   0,   3,
  0, 255,   0,   0,   0,   0,   0,   0, 255,   0,   0,   3,   0,   0,   0,   3,
  0, 255, 255, 255,   0,   0, 255, 255, 255,   0,   0,   3,   3,   3,   3,   3,
  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
240, 240, 240, 240,   0,   0, 255, 255,   0,   0, 255, 255,   0,   0,   0,   0,
240,   0,   0, 240,   0,   0, 255,   0,   0,   0,   0, 255,   0,  32,   0,  32,
240,   0,   0,   0,   0,   0, 255,   0,   0,   0,   0, 255,   0,  32,   0,  32,
240,   0,   0,   0,   0,   0, 255,   0,   0,   0,   0, 255,   0,  32,   0,  32,
240,   0,   0, 240,   0,   0, 255,   0,   0,   0,   0, 255,   0,  32,   0,  32,
240, 240, 240, 240,   0,   0, 255, 255, 255, 255, 255, 255,   0,  32,  32,  32,
```

building

```
  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
  3,   3,   3,   3, 255, 255, 255, 240, 240, 255, 255, 255,   3,   3,   3,   3,
  0,   0,   0,   0,   3,   0,   0,   0,   0,   0,   0,  48,   0,   0,   0,   0,
  0,   0,   0,   0,   3,   0,   0,   0,   0,   0,   0,  48,   0,   0,   0,   0,
  0,   0,   0,   0, 255, 255, 255, 255, 255, 255, 255, 255,   0,   0,   0,   0,
  0,   0,   0,   0, 192,   0,   0,   0,   0,   0,   0,  60,   0,   0,   0,   0,
  0,   0,   0,   0, 192,   0,   0,   0,   0,   0,   0,  60,   0,   0,   0,   0,
  0,   0,   0,   0, 255, 255, 255, 255, 255, 255, 255, 255,   0,   0,   0,   0,
  0,   0,   0,   0, 195,   0,   0,   0,   0,   0,   0,  60,   0,   0,   0,   0,
  0,   0,   0,   0, 195,   0,   0,   0,   0,   0,   0,  60,   0,   0,   0,   0,
  0,   0,   0,   0, 255, 255, 255, 255, 255, 255, 255, 255,   0,   0,   0,   0,
  0,   0,   0,   0, 195,   0,   0,   0,   0,   0,   0,  12,   0,   0,   0,   0,
  0,   0,   0,   0, 195,   0,   0,   0,   0,   0,   0,  12,   0,   0,   0,   0,
 48,  48,  48,  48, 255,   0, 255, 240, 240, 255, 255, 255,  48,  48,  48,  48,
  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
```

snake

```
  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
  0,   0,   0, 255,   0,  12,   0,   0,   0,  48,  48, 255, 255, 255, 255,   0,
  0,   0,   0, 255, 255, 255,   0,   3,   0,   0, 255, 255,   0,   0, 255,   0,
  0,   0,   0,   0,   0, 255,   3,  51, 192,   0, 255,   0,   0, 240, 255,   0,
  0,   0,   3,   3,   3, 255,   0,  48, 192, 195, 255,   3,   0,   0, 255,   0,
  0,   0,  48, 240, 240, 255, 255,  60,   0,   0, 255,   0,  12,  12, 255,   0,
  0,   0,  12, 204,  12, 255,   0,   0,   0, 255, 255,   0,   0,   0, 255,   0,
  0,   0,   3,   3,   3, 255,  12,  12,   0, 255,   0,   0,   0, 255, 255,   0,
  0,   0,   0,   0,   0, 255,  51,  51,   0, 255,  12, 204,   0, 255,   0,   0,
  0,  12,   0, 255, 255, 255,   0,   0,   0, 255, 192, 192,   0, 255, 195,   0,
```

```
  0, 252, 204, 255,   0,  12,   0,   0,   0, 255,   3,  51,  48, 255,   0,   0,
  0, 240,  48, 255,   0, 204, 195,   0, 255, 255,   0,   0,   0, 255,   0,   0,
  0,   3,   3, 255,   0,  48, 195,   0, 255,   0,   0,   0, 255, 255, 195,   0,
  0,   0,   0, 255, 255, 255, 255, 255, 255,  48,   0,   0, 255,  12, 204,   0,
  0,   0,   0,   0,   3, 192,   0,  12,  48,   0,   0, 255, 255,   0,   0,   0,
  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,


tlevel
  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
  0,   0,   0, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,   0,   0,   0,
  0,   0,   0, 255,   0,   0,   0, 255, 255,   0,   0,   0, 255,   0,   0,   0,
  0,   0,   0,   0,   0,   0,   0, 255, 255,   0,   0,   0,   0,   0,   0,   0,
  0,   0,   0,   0,   0,   0,   0, 255, 255,   0,   0,   0,   0,   0,   0,   0,
  0,   0,   0,   0,   0,   0,   0, 255, 255,   0,   0,   0,   0,   0,   0,   0,
  3,   3,  48,  48,   0,   0,   0, 255, 255,   0,   0,   0,  48,  48,   3,   3,
  3,   3,  48,  48,   0,   0,   0, 255, 255,   0,   0,   0,  48,  48,   3,   3,
  0,   0,   0,   0,   0,   0,   0, 255, 255,   0,   0,   0,   0,   0,   0,   0,
  0,   0,   0,   0,   0,   0,   0, 255, 255,   0,   0,   0,   0,   0,   0,   0,
  3,   3, 192, 192,   0,   0,   0, 255, 255,   0,   0,   0, 192, 192,   3,   3,
  3,   3, 192, 192,   0,   0,   0, 255, 255,   0,   0,   0, 192, 192,   3,   3,
  0,   0,   0,   0,   0,   0, 255, 255, 255, 255,   0,   0,   0,   0,   0,   0,
  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,


layers
  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
  0, 255, 255, 255, 255, 255, 255, 204, 204, 255, 255, 255, 255, 255, 255,   0,
  0, 255,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0, 255,   0,
  0, 255,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0, 255,   0,
  0, 255,   0,   0, 255, 255, 255,  80,  80, 255, 255, 255,   0,   0, 255,   0,
  0, 255,   0,   0, 255,   0,   0,   0,   0,   0,   0, 255,   0,   0, 255,   0,
  0, 255,   0,   0, 255,   0,   0,   0,   0,   0,   0, 255,   0,   0, 255,   0,
  0, 243,   0,   0,  65,   0,   0,  21,  21,   0,   0,  65,   0,   0, 243,   0,
  0, 243,   0,   0,  65,   0,   0,  21,  21,   0,   0,  65,   0,   0, 243,   0,
  0, 255,   0,   0, 255,   0,   0,   0,   0,   0,   0, 255,   0,   0, 255,   0,
  0, 255,   0,   0, 255,   0,   0,   0,   0,   0,   0, 255,   0,   0, 255,   0,
  0, 255,   0,   0, 255, 255, 255,  16,  16, 255, 255, 255,   0,   0, 255,   0,
  0, 255,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0, 255,   0,
  0, 255,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0, 255,   0,
  0, 255, 255, 255, 255, 255, 255,  15,  15, 255, 255, 255, 255, 255, 255,   0,
  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,


corners
255, 255,   0,   0,   0, 240, 240, 240, 240, 240, 240,   0,   0,   0, 255, 255,
255,   0,   0,   0,   0, 240,   0,   0,   0,   0, 240,   0,   0,   0,   0, 255,
```

```
  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
  0,   0,   0,   0,   4,  16,   0,   0,   0,   0,  16,   4,   0,   0,   0,   0,
  0,   0,   0,   0,  16, 255, 255,   3,   3, 255, 255,  16,   0,   0,   0,   0,
  3,   3,   0,   0,   0, 255,   0,   0,   0,   0, 255,   0,   0,   0,   3,   3,
  3,   0,   0,   0,   0,  15,   0,   0,   0,   0,  15,   0,   0,   0,   0,   3,
  3,   0,   0,   0,   0,  15,   0,   0,   0,   0,  15,   0,   0,   0,   0,   3,
  3,   3,   0,   0,   0, 255,   0,   0,   0,   0, 255,   0,   0,   0,   3,   3,
  0,   0,   0,   0,  16, 255, 255,  12,  12, 255, 255,  16,   0,   0,   0,   0,
  0,   0,   0,   0,   4,  16,   0,   0,   0,   0,  16,   4,   0,   0,   0,   0,
  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
255,   0,   0,   0,   0, 240,   0,   0,   0,   0, 240,   0,   0,   0,   0, 255,
255, 255,   0,   0,   0, 240, 240, 240, 240, 240, 240,   0,   0,   0, 255, 255,
```

blockades

```
  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
192, 192, 255, 255,  48,   0, 255, 255,   3,   0, 255, 255, 192, 192, 192, 192,
  0,   0, 255, 255,  48,   0, 255, 255,   3,   0, 255, 255, 192,   0,   0,   0,
  0,   0,   0,   0,  48,   0,  12,   0,   3,   0,   0,   0, 192,   0,   0,   0,
  0,   0,   0,   0,  48,   0,  12,   0,   3,   0,   0,   0, 192,   0,   0,   0,
  0,   0,   0, 255, 255,  48,  12, 255, 255, 192,  12, 255, 255,   3,   3,   3,
  0,   0,   0, 255, 255,  48,  12, 255, 255, 192,  12, 255, 255,   0,   0,   0,
  0,   0,   0,   0,   3,  48,   0,   0,   0, 192,  12,   0,   0,   0,   0,   0,
  0,   0,   0,   0,   3,  48,   0,   0,   0, 192,  12,   0,   0,   0,   0,   0,
  3,   3, 255, 255,   3,   0, 255, 255,  60,  12, 255, 255,  12,   0,   0,   0,
  0,   0, 255, 255,   3,   0, 255, 255,  48,   0, 255, 255,  12,   0,   0,   0,
  0,   0,  12,   0,   0,   0, 192,   0,  48,   0,   3,   0,  12,   0,   0,   0,
  0,   0,  12,   0,   0,   0, 192,   0,  48,   0,   3,   0,  12,   0,   0,   0,
 48,  48,  60, 255, 255,   0, 192, 255, 255, 192, 195, 255, 255,  48,  48,  48,
  0,   0,  12, 255, 255,   3, 195, 255, 255,   0,   3, 255, 255,   0,   0,   0,
  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
```

tribuilding

```
  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
 15,  15,  15, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 240, 240, 240,
 15,   0,   0, 255,   0,   4,   0,   0,   0,   0,   4,   0, 255,   0,   0, 240,
 15,   0,   0, 255, 128,   0,   0,   0,   0,   0,   0, 128, 255,   0,   0, 240,
 15,  15,   0, 255,   0,   4,   0,   0,   0,   0,   4,   0, 255,   0, 240, 240,
  0,   0,   0, 255, 255, 255, 255,   0,   0, 255, 255, 255, 255,   0,   0,   0,
  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
255, 255, 255,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0, 255, 255, 255,
255,   0, 255,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0, 255,   0, 255,
255,   0,   0,   0, 255, 255,   0,   0,   0,   0, 255, 255,   0,   0,   0, 255,
255,  32,   0,   0,   0, 255,   0,   0,   0,   0, 255,   0,   0,   0,  32, 255,
255,   0,   0,   8,   0, 255,   0,   0,   0,   0, 255,   0,   8,   0,   0, 255,
255, 255, 255, 255, 255, 255,   0,   0,   0,   0, 255, 255, 255, 255, 255, 255,
```

```
  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
```

rooms

```
  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
  0, 255, 255, 255, 255, 255,   0,   0,   0,  15,  15,  15,  15,  15,  15,   0,
  0, 255,   0,   0,   0, 255,   0,   0,   0,  15,   8,   0,   0,   8,  15,   0,
  0, 255,   0,  72,   0,   0,   0, 130,   0,  15,   0,   1,   0,   0,  15,   0,
  0, 255,   0,  72,   0,   0,   0, 130,   0, 255, 242, 242, 240, 240,  15,   0,
  0, 255,   0,   0,   0, 255,   0, 130,   0, 255,   0,   1,   0, 240,  15,   0,
  0, 255, 255, 255, 255, 255,   0,   0,   0, 255,   0,  36,  38,   2,  15,   0,
  0,   0,   0,   0,   0,   0,   0,   0,   0, 255,   0,  37,  36,   0,  15,   0,
  0,   0,   0,   0,   0,   0,  15,  15,  15, 255,   0,   0,   0,   0,  15,   0,
  0,   0,   0,   0,   0,   0,  15,   8,   2, 240, 240, 241, 240, 240,  15,   0,
  0, 255, 255, 255, 255,   0,  15,   0,   2,   4,   4,   0,   0,   0,  15,   0,
  0, 255,   0,   0, 255,   0,   0,   0,   0, 245, 244, 241, 242,   0,  15,   0,
  0, 255,   0,   0, 255,   0,  15,   8,   0,   0,   0,   0, 242,   8,  15,   0,
  0, 255,   0, 255, 255,   0,  15,  15,  15,  15,   0,   0, 255,  15,  15,   0,
  0,   0,   0,   0,   0,   0,   0,   0,   0, 240, 240, 240, 240,   0,   0,   0,
  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
```

happy

```
  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
  0,   0,   0,   0, 255, 255, 255,  15,  15, 255, 255, 255,   0,   0,   0,   0,
  0,   0,   0, 255,   0,   0,   0,   0,   0,   0,   0,   0, 255,   0,   0,   0,
  0,   0, 255,   0,   3,   0,   0,   0,   0,   0,   0,   3,   0, 255,   0,   0,
  0, 255,   0,   0,   0,   3,   0,   0,   0,   0,   3,   0,   0,   0, 255,   0,
  0, 255,   0,   0, 255, 255,   3,   0,   0,   3, 255, 255,   0,   0, 255,   0,
  0, 255,   0,   0, 255, 255,   0,   0,   0,   0, 255, 255,   0,   0, 255,   0,
  0, 255,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0, 255,   0,
  0, 255,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0, 255,   0,
  0, 255,   0,   0,  32,  32,  32,  32,  32,  32,  32,  32,   0,   0, 255,   0,
  0, 255,   0,   0, 255,  32,  32,  32,  32,  32,  32, 255,   0,   0, 255,   0,
  0,   0, 255,   0,   0, 255, 255, 255, 255, 255, 255,   0,   0, 255,   0,   0,
  0,   0,   0, 255,   0,   0,   0,   0,   0,   0,   0,   0, 255,   0,   0,   0,
  0,   0,   0,   0, 255, 255, 255, 240, 240, 255, 255, 255,   0,   0,   0,   0,
  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
```

palace2

```
  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
255, 255, 255, 255, 255, 255,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
255,   0,   0,   0,   0, 255,  15,  15, 255, 255, 255, 255, 255, 255, 255,   0,
255,   0,   0,   0,   0, 255,   0,   0,   0,   0, 255,   0,   0,   0, 255,   0,
255,   0,   0,   0,   0, 255,   0,  64,  64,   0, 255,   0,   0,   0, 255,   0,
```

```
255,   0,   0,   0,   0,   0,   0,  64,  64,   0,  15,   0,   0,   0, 255,   0,
255,   0,   0,   0,   0,   0,   0,   0,   0,   0,  15, 240, 255, 255, 255,   0,
255, 255, 255, 255, 255,   0,   0,   0,   0,   0,   0,   0,   0,   0, 255,   0,
255,   0,   0,   0,   0,   0,  15,  15,  15, 255, 255,   0,   0,   0, 255,   0,
255,   0,   0, 240,   0,   0,   0,  15,   0, 240,   0,   0, 255, 255, 255,   0,
255,   0,   0, 240,   0,   0,   0,  15,   0, 240,   0,   0, 240,   0,   0,   0,
255,   0,   0, 240,   0,   0,   0,  15,   0, 240,   0,   0, 240,   0,   0,   0,
255, 255, 255, 255,  15,  15, 255, 255, 255, 255, 255, 255, 255,   0,   0,   0,
  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
```

trails

```
  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
  0,   0,   0,   0,   0,  64,  64,  64,  64,   0,   0,   0,   0,   0,   0,   0,
  0,   0, 255, 255, 255, 255,  17,  17,  65,   1,   1,   0,   0,   0,   0,   0,
  0,   0, 255, 255, 255, 255,   0,  16,  80,   0,   1,   0,   0,   0,   0,   0,
  0,   8, 255, 255, 255, 255,   4,   4,  84,  20,   5,   4,   4,   4,   0,   0,
  0,   8,  32,   0,   2,   0,   0,   0,  64,  16,  17,  16,   0,   4,   0,   0,
  0,   8,  32,   0,   2,   0,   0,   0,  64,   0,   1,  16,   0,   4,   0,   0,
  0,   8,  32,  32,  34,  32,  32,  32,  96, 255, 255, 255, 255,   4,   0,   0,
  0,   8,  32,   0,   2,   2,   2,   2,  66, 255, 255, 255, 255,   0,   0,   0,
  0,   8,  32,   0,   2,   4,   4,   4,  68, 255, 255, 255, 255,   0,   0,   0,
  0,   8,  32,   0,   2,   4,   0,   0,  64,  16,   1,   0,   8,   0,   0,   0,
  0,   8,  32,   0,   2,   4,   0,   0,  80,  16,   1,   0,   8,   0,   0,   0,
  0,   8, 255, 255, 255, 255,  65,  81,  81,   1,   1,   0,   8,   0,   0,   0,
  0,   0, 255, 255, 255, 255,  16,  16,   0,   0,   0,   0,   8,   0,   0,   0,
  0,   0, 255, 255, 255, 255,   8,   8,   8,   8,   8,   8,   8,   0,   0,   0,
  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
```

scattershot

```
 64,   4, 130,   0,   8,   0,  16,   0,   0, 128,  20,  32,  65,   0,   8,  64,
  8, 145,   0,  52,  64, 128,   2,  16,   3,   0,  40,   0,  66,  16,   1,   8,
  0,  32,  68,   0, 149,  34, 128,   8, 192,   4, 128,   8,  16, 128,  17,   0,
  0,  10,   0,   2,   0,   0,   0,  33,  72,  18,   1,  65,  48,   2, 128,  64,
  0, 192,  64,  25, 132,   0,  36,   0,  48, 192,   8,   0,   0,   8,   1,   0,
 32,   0, 130,  36,  16,   8,   0, 128,  13, 128, 128, 132,  96,   4,   0,  36,
  0,  88,   1,   0,  65, 134,   0,   2,   0,  16,  10,  96,   0,   3,   0,   0,
  0,  33,   0,  32,  88,   2,  96,  16,  32,   4,   1,   4,   0, 128,  24,  33,
128,   0, 148,   0,   1,   0,   1, 128,   8,   0,   8,   0,  80,   6,   2,   0,
 16,  36,   4,   2, 104,  16,   0,  36,   0,  64,  66, 128,  32, 128,   0, 128,
  0,   0,  80,  96, 128,   9,   0,  16,  34,   1,   4,  16,   1,  32,  36,  16,
  0,   2,   0,   4,   8,   0,   0, 132,  64,   8,  66,   0,   6,  89,   0,   0,
  0,  96,  33, 145,  66,  16,  80,  32,  16, 161,   0, 160, 128,  32,   0,  16,
128,  76,   0,   2,  16,   6,   0,   0,  11,   0,   8,   1,   0,   4,  10,  64,
  0,   1,  32,   8,  33,   8, 128,  16,   4,   0,  64,  32,  80,   3,   0,   0,
128,   0,  16, 128,   0,   0,   4,   0,  64,  32,   4, 128,   0, 128,  16,  64,
```

```
rooms3
255, 255, 255,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
255,   0,   0,   0,   0,   0,   0,   0,  15,  15,  15,  15,  15,   0,   0,   0,
255,   0,   0,   0,   0,   0,   0,   0,  15,   0,   0,   0,  79,  64,  64,  64,
255, 255, 255,   0,   0,   0,   0,   0,   0,   0,   0,   4,  79,   0,   0,  64,
255,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  79,   0,   0,  64,
255,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   4,  79,   0,   0,  64,
255, 255, 255,   0,   0,   0,   0,   0,   0,   0,   0,   0,  15,   0,  64,  64,
255,   0,   0,   0,   0,   0,   0,   0,  15,   0,   0,   0,  15,   0,   0,   0,
255,   0,   0,   0,   0,   0,   0,   0,  15,  15,  15,  15,  15,   0,   0,   0,
255, 255, 255,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
255,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
255,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
255, 255, 255,   0,   0,   0, 240, 240, 240,   0,   0,   0, 240, 240, 240,   0,
255,   0,   0,   0,   0,   0, 240,   0,  64,   0,   0,   0,  64,   0, 240,   0,
255,   0,   0,   0,   0,   0, 240,   0,   0,   0,   0,   0,   0,   0, 240,   0,
255, 255, 255,   0,   0,   0, 240, 240, 240, 240, 240, 240, 240, 240, 240,   0,


rooms4
255, 255,   0, 255, 255,   0, 255, 255, 255, 255,   0, 255, 255,   0, 255, 255,
255,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0, 255,
  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
  3,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   3,
  3,   0,   0,  15, 255, 255, 255, 240, 240,   0,   0,   0,  15,   0,   0,   3,
  0,   0,   0,  15,   0,   0,   0, 240, 240,   0,   0,   0,  15,   0,   0,   0,
  3,   0,   0,  15,   0,   0,   0, 240, 240,   0,   0,   0,  15,   0,   0,   3,
  3,   0,   0,  15,   0,   0,   0, 240, 240,   0,   0,   0,  15,   0,   0,   3,
  0,   0,   0,  15,   0,   0,   0, 240, 240, 255, 255, 255,  15,   0,   0,   0,
 48,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  48,
 48,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  48,
  0,   0, 255, 240, 240, 240,   0,   0,   0,   0, 240, 240, 240, 255,   0,   0,
 48,   0, 255,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0, 255,   0,  48,
 48,   0, 255,   0,   0,   0,  15,  15,  15,  15,   0,   0,   0, 255,   0,  48,
  0,   0, 255,   0,   0,   0,  15,  15,  15,  15,   0,   0,   0, 255,   0,   0,
  0,   0, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,   0,   0,


rooms5
255, 255, 255, 255, 255, 255,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
255,   0,   0,   0,   0, 255,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
255,   0,   0,   0,   0,   0,   0,   0,   0,  15,  15,  15,  15,  15,   0,   0,
255,   0,   0,   0,   0,   0,   0,   0,   0,  15,   0,   0,   0,  15,   0,   0,
255,   0,   0,   0,   0,   0,   0,   0,   0,  12,   0,   0,   0,   3,   0,   0,
255,   0,   0,   0,   0, 255,   0,   0,   0,  15,   0,   0,   0,  15,   0,   0,
255, 255, 255, 255, 255, 255,   0,   0,   0,  15,  15,  15,  15,  15,   0,   0,
  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
```

```
 16,  16,  16,  16,   0,   0,  64,  64,  64,  64,   0,   0,  16,  16,  16,  16,
  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
  0,   0,  15,  15,  12,  15,  15,   0,   0,   0, 255, 255, 255, 255, 255, 255,
  0,   0,  15,   0,   0,   0,  15,   0,   0,   0, 255,   0,   0,   0,   0, 255,
  0,   0,  15,   0,   0,   0,   3,   0,   0,   0,   0,   0,   0,   0,   0, 255,
  0,   0,  15,   0,   0,   0,  15,   0,   0,   0, 255,   0,   0,   0,   0, 255,
  0,   0,  15,  15,  15,  15,  15,   0,   0,   0, 255, 255, 255, 255, 255, 255,
```

rooms6

```
 16,  16, 255, 255, 255, 255,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
 16,   0, 255,   0,   0, 255, 255,   0,   0,   0,   0,   0,   0,   0,   0,   0,
  0,   0, 255,   0,   0,   0, 255,  15,   0,   0, 255, 255, 255, 255, 255, 255,
  0,   0, 255,   0,   0,   0,   0,  15,   0,   0,  15,   0,   0,   0,   0, 255,
  0,   0, 240,   0,  64,   0,   0,   0,   0,   0,  15,   0,   0,   4,  15, 255,
  0,   0, 240,   0,  64,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0, 255,
  0,   0, 255,   0,   0,   0,   0,  15,   0,   0, 240,   0,   0,   4,  15, 255,
  0,   0, 255,   0,   0,   0, 255,  15,   0,   0, 240,   0,   0,   0,   0, 255,
  0,   0, 255,   0,   0, 255, 255,   0,   0,   0, 255, 255, 255, 255, 255, 255,
  0,   0, 255, 255, 255, 255,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
 15,  15,  15,  15,  15,   0,   0,   0, 255, 255,   0,   0,   0, 255, 255,   0,
 15,   0,   0,   0,   0,   0,   0, 255, 255,   0,   0,   0,   0,   0, 255, 255,
 15,   4,   4,   0,   0,   0,   0, 255,   0,   0,  48,  48,  48,   0,   0, 255,
 15,   0,   0,   0,  15,   0,   0, 255,   0,   0,   0,   0,   0,   0,   0, 255,
 15,  15,  15,  15,  15,   0,   0, 255, 255, 255, 255, 255, 255, 255, 255, 255,
```

rooms7

```
 15,  15,  15,  15,   0, 240, 240, 240, 240,   0,   0,  15,  15,  15,  15,  15,
 15,   0,   0,  15,   0, 240,   0,   0, 240,   0,   0,  15,   0,   0,   0,  15,
 15,   1,   0,   0,   0, 240,   0,   0, 240,   0,   0,   0,   0,   0,   4,  15,
 15,   0,   0,   0,   0, 240,   0,   0, 240,   0,   0,   0,   0,   0,   4,  15,
 15,   0,   0,  15,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   4,  15,
 15,  15,  15,  15,   0,   0,   0,   0,   0,   0,   0,  15,   0,   0,   0,  15,
  0,   0,   0,   0,   0,   0,   0, 255, 255,   0,   0,  15,  15,  15,  15,  15,
240, 240, 240, 240, 240,   0, 255,   0,   0, 255,   0,   0,   0,   0,   0,   0,
240,  64,   0,   0,   0,   0, 255,   0,   0, 255,   0,   0,   0,   0,   0,   0,
240,   0,   0,   0,   0,   0,   0, 255, 255,   0,   0,   0,  15,  15,  15,  15,
240, 240, 240, 240, 240,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  15,
  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  15,
 15,  15,  15,  15,  15,   0,   0,  15,  15,  15,   0,   0,  15,   0,   8,  15,
 15,   0,   0,   0,   0,   0,   0,   0,   0,  15,   0,   0,  15,   0,   0,  15,
 15,   0,   8,   0,   8,   0,   0,   0,   0,  15,   0,   0,  15,   0,   0,  15,
 15,  15,  15,  15,  15,  15,  15,  15,  15,  15,   0,   0,  15,  15,  15,  15,
```