Case Study of Room Generation System:

PROBLEMS AND SOLUTIONS
SUMMARY OF SYSTEM
DESIGN TENETS
PROCESS
DEMO
RUBRIC
ATTRIBUTES
FINAL WALKTHROUGH

Problems and solutions

PROBLEM(S):

Procedural generation is soulless and bland

Pure randomness does not equate to good gameplay

Unbalanced rooms or item spawns

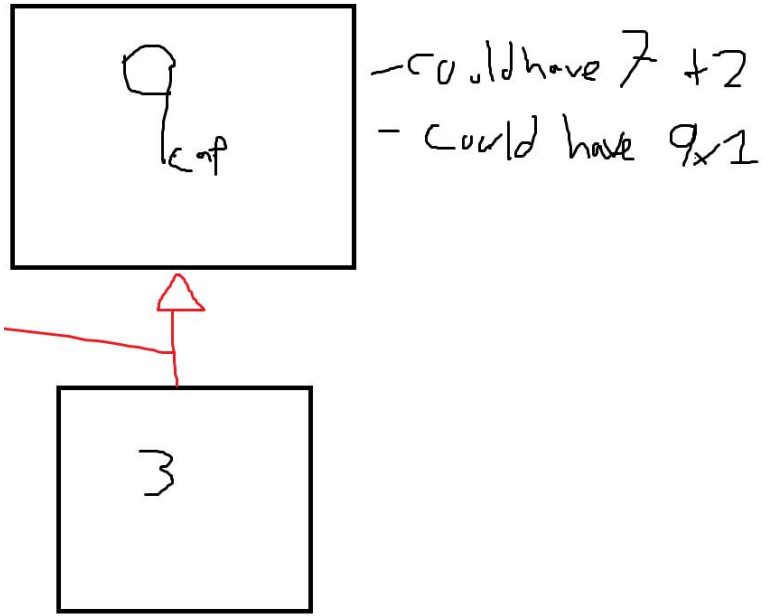
SOLUTION:

A system that uses a numerical formula to determine the next room in the sequence based on its point value. The value would also determine what is allowed to spawn in the room.

Summary

DESCRIPTION:

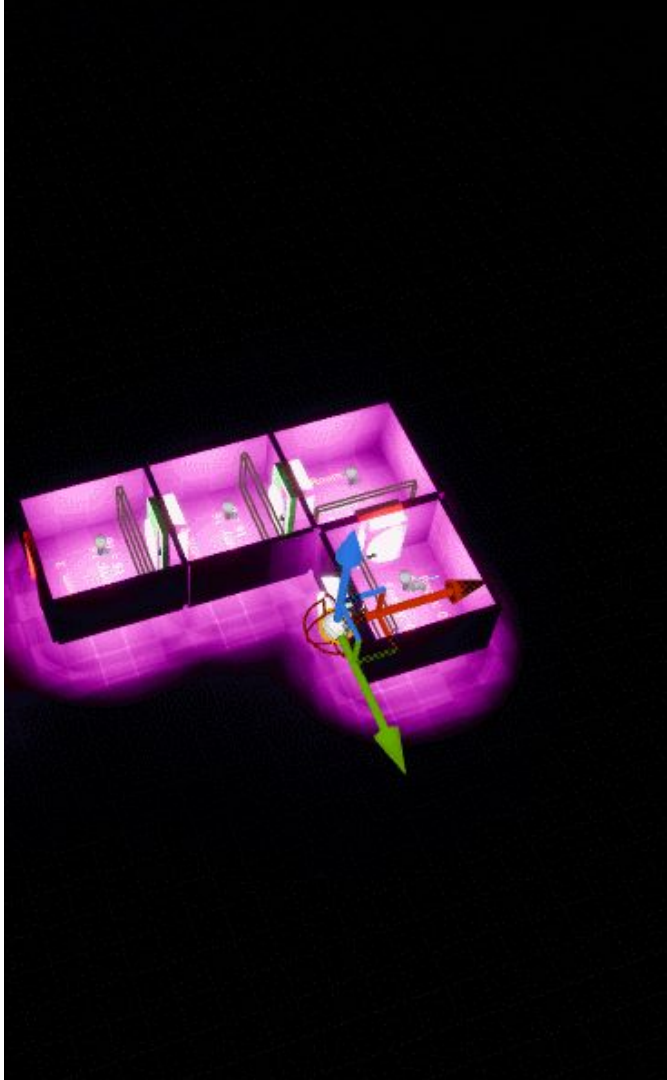
This would be a room generation system that would not only spawn the next room in a sequence but fill the room based on the budget allocated to it.



Ref: MS Paint sketch

Design Tenets

BUDGET INTEGRITY
CONSISTENCY ACROSS ROOMS
HISTORICAL STABILITY
SCALABLE DIFFICULTY
BALANCED CONTENT DISTRIBUTION



Inputs

- RAMP UP VALUE
- NUMBER OF ROOMS
- HISTORICAL TRENDS
- ENEMY COST
- HAZARD COST
- STARTING BUDGET

Process

- ROOM GENERATION
- ITEM PURCHASES
- PURCHASES CHECK
- OPTIMIZATION
- LAYOUT

Outputs

- ITEM SPAWNING
- GRADED RUBRIC
- VISUAL DUNGEON LAYOUT
- HISTORICAL DATA

DEMO

LINK

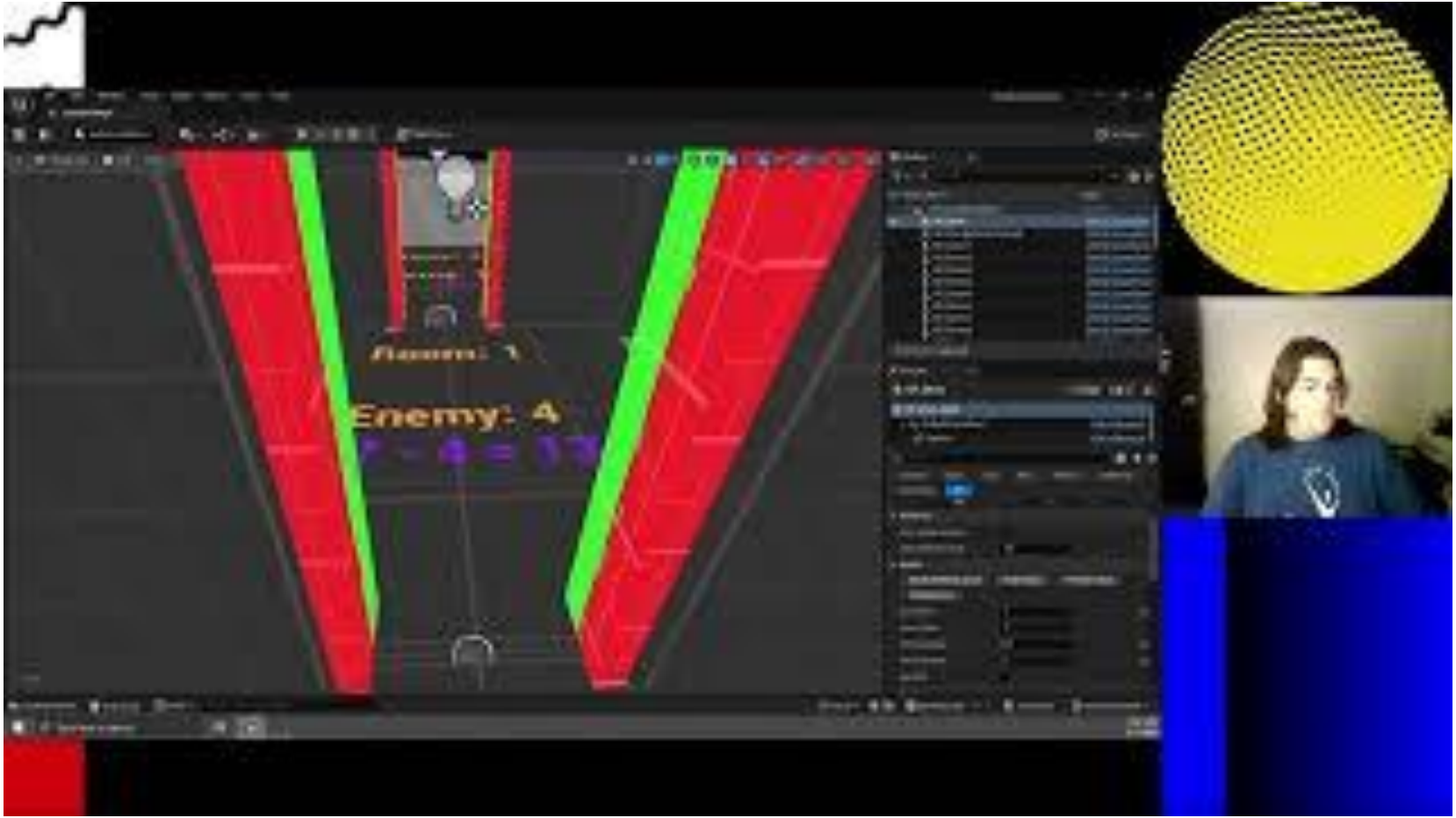
This demo was made in processing as a initial proof of concept for the system

Rubric

CRITERIA:	A	B	C	D	F
Enemies	Enemy counts in every room match the design target exactly (error ≤ 0.2; expected = roomBudget/6).	Enemy counts are mostly as expected (error ≤ 0.4; expected = roomBudget/6)	Enemy counts sometimes deviate (error ≤ 0.6; expected = roomBudget/6)	Enemy counts frequently stray (error ≤ 1.0; expected = roomBudget/6)	Enemy counts are consistently off-target (error > 1.0; expected = roomBudget/6)
Hazards	Hazard counts are perfectly proportional (error ≤ 0.2; expected = roomBudget/2.5)	Hazard counts are generally on target (error ≤ 0.4; expected = roomBudget/2.5)	Hazard counts occasionally deviate (error ≤ 0.6; expected = roomBudget/2.5)	Hazard counts frequently miss the mark (error ≤ 1.0; expected = roomBudget/2.5)	Hazard counts are completely off (error > 1.0; expected = roomBudget/2.5)
Room Value	Room budgets are highly consistent (variation ratio ≤ 0.3)	Room budgets are consistent (variation ratio ≤ 0.4)	Room budgets show moderate variation (ratio ≤ 0.5)	Room budgets vary considerably (ratio ≤ 0.7)	Room budgets are highly inconsistent (ratio > 0.7)
Dungeon Value	Total dungeon value is within 15% of ideal (ideal = sum[startingBudget + (i-1)*rampUpValue])	Total dungeon value is within 20% of ideal	Total dungeon value is within 25% of ideal	Total dungeon value is within 30% of ideal	Total dungeon value deviates by more than 30% from ideal
Consistency	Current total room value is within 15% of the regression prediction	Within 20% of the regression prediction	Within 25% of the regression prediction	Within 30% of the regression prediction	Differs by more than 30% from the regression prediction
Historical	Current enemy/hazard ratios differ by ≤ 15% from historical averages	Differ by ≤ 20%	Differ by ≤ 25%	Differ by ≤ 30%	Differ by more than 30% from historical averages

	ATTRIBUTES	INTERACTIONS
Room	• **roomNumber**: Identifier for the room • **roomBudget**: Total points available for items • **currentPoints**: Points used so far • **enemyCount**: Count of enemy tokens • **hazardCount**: Count of hazard tokens	• Acts as the primary container; each room "buys" items (enemies and hazards) using its point budget. • Its budget is determined by a starting value and ramp-up (with random variation). • Provides aggregate data (room value, enemy/hazard counts) used in scoring and history tracking. • Connected via hallways to form a dungeon layout.
Enemy	• **Level**: Indicates the enemy's strength and point cost (allowed values: 4, 5, 6, 7, 8) • **Type**: A descriptive string (e.g., "Level 7 Enemy")	• Each enemy token "costs" a certain number of points; it is "purchased" within a room if there is enough remaining budget. • The number and cost of enemies in a room affect balance and are evaluated by the rubric (expected enemy count is derived from roomBudget/average enemy cost).
Hazard	• **Level**: Indicates the hazard's challenge and point cost (allowed values: 1, 2, 3, 4) • **Type**: A descriptive string (e.g., "Level 3 Hazard")	• Each hazard token also "costs" points and is added to a room based on remaining budget. • The ratio of hazards to enemies is important for gameplay balance and is evaluated in the rubric. • When budgets are high, the system tends to prefer higher-level hazards to use points more efficiently.
Points	• **Budget**: A numerical value representing the available points for a Room	• Points are the currency used in each room. The room's budget is "spent" on purchasing enemy and hazard tokens. • The remaining points (roomBudget – currentPoints) determine if additional items can be "bought" or if the system should try to purchase a higher-value token.
Dungeon	• A collection (ArrayList) of Room tokens • **Overall metrics**: Aggregated values (total room value, total enemy count, total hazard count)	• Serves as the overarching container that holds all room tokens. • Provides aggregate data for evaluating the system via the rubric and tracking history. • The dungeon's overall layout (rooms connected by hallways) forms the complete game area.

WALKTHROUGH



LINK