ECS7003P Multi-platform Game Development

Resit Assignment Submission

# Meta information

Student name: Cuong Minh Phan

Git repository URL: https://github.com/jakelong12556/ToxicDungeons

Gameplay video URL: https://youtu.be/gdEV8BPBwR4

# Game information

Game concept chosen: # 2

Game title: Toxic Dungeon

Game concept changes (if any; **not** limited to 2!):

* The game environment is changed to a dungeon.
* The player must kill all enemies in a level for the dungeon to reveal the exit.
* The dungeon is most unkind, it changes shapes when the player reaches closer to the exit.
* Poison is used instead of hunger, and thus requires curing potions instead of food. Hence the name, Toxic Dungeon.

Implemented features: (include all; **not** limited to 2!)

* Environment. Mark expected given marking scheme (out of 12%):
  + Is procedurally generated, given the parameters can be very varied in size and shape.
  + A level manager is used for the specification of the level; this is to ensure balance and challenge. It starts of simple with less enemies and becomes harder as time passes.
  + Three distinct different type of dungeon generators are used to ensure variability. A single room generator, multiple room generator and advanced dungeon generator.
  + Single room generator can be compact or very large to have a large interaction space.
  + Multiple room generators have small single rooms connected by a corridor to make a dungeon.
  + Advanced dungeon generator creates a full dungeon given a specified dungeon size and partitions it to create a more meaningful dungeon.
* Gameplay. Mark expected given marking scheme (out of 36%):
  + Poison ticks away at the players health. They must deal with enemies efficiently to survive both the enemies and the poison.
  + There are three types of enemies in the dungeon. Toxic knight, archer, and mage. They all challenge the player skill at dodging and attacking, and resource management.
  + The knight charges at the player, they must be shot twice to die.
  + The archer fires arrows quickly at the player, but the players bullet can destroy the arrows. They must be shot 3 times to die.
  + The mage fires seeking toxic balls at a slower rate. But the player must shoot the balls down. They are fragile and die in 1 shot.
  + Lamps illuminate the dungeon; they also act as obstacles for the both the enemy and player to hide behind. When the fighting gets rough, hide behind cover. One type of lamp is stationary, the other can be pushed.
  + The enemies only chase the player within their range inside of their rooms. This ensures the player being in a big dungeon does not get swarmed.
  + The shooters stay at range, but if the player retreats, they will close in and chase the player.
  + As the game revolves around tight spaces, the player can shoot and move at the same time. This is to effectively dodge attacks.
* Non-playable. Mark expected given marking scheme (out of 12%):
  + The player can adjust the music volume in the option menu. This setting is saved in a JSON based txt file.
  + The player is given access to a menu on entry, as well as a pause menu in game.
  + HP and Toxic Resistance are defined as red and green bars. If either depletes to 0 the player dies and game ends.
  + When hovering over the exit ladder, a popup indicates the player can press the enter key to move onto the next level.
  + A level transition is used to ensure the player gets a bit of time to plan their attack in the new level and not be surprised by enemies.
* Aesthetics. Mark expected given marking scheme (out of 12%):
  + Appropriate dungeon music is used to set the atmosphere. It rushes the player as they do not have much time, given the poison.
  + The camera follows the player in the centre through level transitions.
  + When the player is hit, the screen shakes, the player flashes red. The screen shakes as it is the player.
  + When the enemy is hit, they flash red to indicate being hit.
  + When the entities die, they explore into a red mist.

# Research

Implementation of PCG in Toxic Dungeons

Research implemented features:

PCG – Level manager and levels.  
PCG – Random Walk, Binary Space Partitioning

The usage of PCG in Toxic Dungeons is motivated by several benefits such as being used to control game difficulty/balancing, reduce development time, and increase replay ability (Breno and Selan, 2019). As such, the development of the dungeon generators in this game makes use of the taxonomy of PCG for its game design as given by (Shaker et al. 2016) and redefined by (Breno and Selan, 2019). The taxonomy defines 7 axes to classify the approaches to PCG in dungeons, of which Toxic Dungeons defines below in its design.

1. *Content need:* the dungeons generated is **non-optional**.
2. *Generation time:* the dungeon is generated before and in-between active gameplay, such as before starting the game and in-between levels. Hence, **offline** generation.
3. *Generation control:* the dungeon is controlled by **parameters** that defines the rooms shapes, room count and overall dungeon shape.
4. *Generality*: the dungeon is not adaptive to player performance, and thus is **generic**.
5. *Random choice:* Given the same parameters, the dungeon generation is different every new run. Hence it is a **stochastic** approach.
6. *Generation Method*: The dungeon is generated in one pass, and thus makes use of a **constructive** approach.
7. *Content Authorship*: As the designer only has controls over the parameters of the dungeon, the computer generates the content primarily and as such is of type **automatic generation.**

These approaches allow Toxic Dungeon to make use of a clearly defined level manager to structure the levels from easy to hard. The parameters also allow for adjustments in enemy count and item count to properly populate a room given its size. Finally, with the stochastic, single pass approach, level generation does not require the designer development and thus more focus can be allocated on other aspects.

Histogram

Description automatically generated The two main implemented algorithm in dungeon generation are Random Walk (Baron, 2017) and BSP (Shaker et al. 2016) (Baron, 2017). Random Walk/Drunkards Walk is a cellular algorithm that walks randomly in the 4 cardinal directions to create the shape a dungeon room. Parameters such as walk length and walk iterations increase the walk area and thus the room size (Baron, 2017). Additionally, random starts are also implemented which changes the starting position of a walk, further increasing a room size and its randomizable. If one repeats the action of a random walk, corridors can be generated. This allows the creation of rooms at the end of each corridor and connect them, to generate a multi-room dungeon (Baron, 2017) as shown in figure 1.

Figure Connected random walk rooms

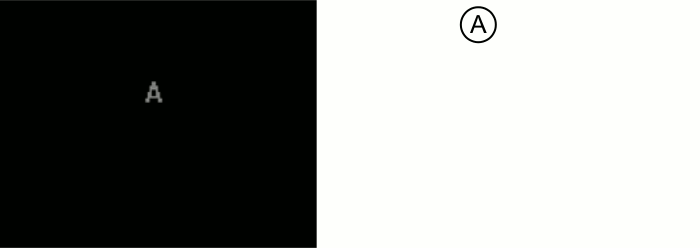
However, the problem with this approach is that the rooms end up being the same rough size and it is less interesting to traverse. Thus, BSP is used to give more variability in room sizes and improve dungeon exploration. Toxic Dungeon makes use of the pseudocode for a stochastic BSP generator as shown below (Shaker et al. 2016).

Text

Description automatically generated

Figure Stochastic BSP pseudocode

This approach is also used as it keeps the rooms from overlapping due to its partitioning approach (Shaker et al. 2016). Below in figure 3 is a visualization of the binary space tree which results in a dungeon. The implementation in the game results in figure 4 which shows a few sample levels generated by the BSP algorithm. As shown, the partitioned rooms achieve a more varied sizes, allowing for different challenges in rooms due to the constraints of space and interesting shapes.



A picture containing text, circuit

Description automatically generatedHistogram

Description automatically generated with medium confidenceA picture containing text, circuit

Description automatically generatedFigure 3 Visualization of stochastic partition for a Dungeon of dimension A.

Figure 4 Sample BSP dungeon generation

# References:

Baron, Jessica. (2017). Procedural Dungeon Generation Analysis and Adaptation. 168-171. 10.1145/3077286.3077566.

Gillian Smith, et al. “Understanding Procedural Content Generation: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems.” *ACM Conferences*, 1 Apr. 2014, https://dl.acm.org/doi/10.1145/2556288.2557341.

Shaker, Noor, et al. Chapter 3, “Procedural Content Generation in Games.” *SpringerLink*, Springer International Publishing, 2016, https://link.springer.com/book/10.1007/978-3-319-42716-4. https://www.antoniosliapis.com/articles/pcgbook\_dungeons.php

Viana, Breno M. F, and Selan R. dos Santos. “A Survey of Procedural Dungeon Generation - SBGames.” *SBGames*, 28 Oct. 2019, https://www.sbgames.org/sbgames2019/files/papers/ComputacaoFull/198359.pdf.

Mark expected given marking scheme (out of 20%):