Name: Sanad Masannat CCID: sanad Date: 11/26/2021

Instructions: Each question is worth the given amount of points, and the whole quiz is worth nine points in total. Answer each question to the best of your ability. Read all instructions carefully. Submit your quiz to eClass before Sunday at 11am. You may submit your quiz as a pdf, a docx, or as a zip file of images. **It is your responsibility to ensure the TAs and instructor can read your answers**, if you’re concerned about that please type your answers when possible.

**Multiple Choice (0.5 points each)**. Circle the answer closest to the one you would give.

**Q1 (0.5 points).** Which of the following PCGML approaches requires the least amount of data?

**(A.)** Bayes Net (B.) Autoencoder

(C.) GAN (D.) LSTM

**Q2 (0.5 points).** Which of the following is a drawback of generative grammars?

(A.) Slow in comparison to constraint solvers or search systems

(B.) Hard to debug(C.) Unintuitive for designers

**(D.)** Worse at making game bits and quests than noise

**Q3 (0.5 points).** When, as a designer, does it make the most sense to incorporate procedural content generation (PCG)?

(A.) When you want all players to have the same, consistent experience

(B.) When you want to fill a large space with unique, high-quality content

(C.) When you want to save development costs

**(D.)** When you want to create a game experience that’s different every time

**Q4 (0.5 points)**. Which of the following is a benefit of search-based PCG approaches?

(A.) Can more easily be implemented to automatically adapt content to a player.

**(B.)** Can guarantee aspects of generated content (like that a level is completable).

(C.) They are typically the fastest PCG approach.

(D.) None of the above are actual benefits of search-based PCG.

**Q5 (2.5 points).** Answer the below parts to the best of your ability.

**Q5.A. (0.5 points**). Design an example dungeon room that will serve as input to a Wave Function Collapse implementation. The game level will be a top-down adventure game and can have the following tokens: blocks (B), keys (K), locked doors (L), monsters (M), the player’s starting position (P), and empty tokens ( ). Your dungeon room must have **exactly 1 player** and must have **at least 1 of all of the other tokens**. In addition, it should be possible to navigate from the player’s starting position to a key and then to a locked door through empty tokens ( ). However, the player’s starting position, keys, and locked doors **must not be neighbours**.

For example, the below would be a legal input (do not answer with anything within 5 changes of this):

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| P |  | K |  | L |
|  |  |  |  |  |
| B |  | M |  |  |
|  |  |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | **B** |  |  |
| **P** |  | **M** |  | **L** |
|  |  |  |  | **M** |
|  | **B** | **K** | **B** |  |

**Q5.B. (2 points)** Give 4 distinct outputs (**0.5 points each**) using WFC on your input given above. Assume we are using a window size of 2x2. Assume we treat each “offscreen” cell/slot as if it is filled with every possible token value simultaneously. Give three unique, valid outputs. Unique means that I shouldn’t be able to exactly overlay any of your outputs on top of one another with any offset. Valid means according to the constraints WFC identifies from your input, the outputs doesn’t have to follow the rules from **Q5A** (e.g. at least one of each token). Yes, it’s intentional that the outputs are different sizes. If you believe that your input doesn’t allow for four unique outputs you may explain why in text, but it’s highly unlikely that, that’s the case.

Output 1 (3x3):

|  |  |  |
| --- | --- | --- |
| P |  | L |
|  |  | M |
| K | B |  |

Output 2 (4x3):

|  |  |  |  |
| --- | --- | --- | --- |
| L |  | M |  |
| M | B | K | B |
|  | M |  | M |

Output 3 (5x4):

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | B |  | P | P |
|  | M |  |  |  |
|  |  |  |  | M |
|  | B | K |  |  |

Output 4 (6x5):

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| L | P |  |  |  |  |
| M |  |  | B | K |  |
|  |  | L |  |  |  |
| K |  | M |  | B | K |
| B | K |  |  | M |  |

**Q6 (1 point)** Imagine you have been tasked to develop the PCG systems for an open world fantasy game. Give the PCG method (discussed in class) you would suggest for each of the following and a brief explanation as to why. (A) The mountainous landmass the player will adventure on, generated based on minimal designer input (**0.25 points**), (B) Villagers and enemies, which must feel hand-crafted (**0. 25 points**). (C) Dungeons for the game, which must be generated quickly at runtime based on designer specifications (**0. 25 points**), and (D) Quests for the game, which must be generated quickly and must involve the previously generated elements (**0. 25 points**). Make sure to give brief (~1 sentence) justifications for each answer.

1. **Noise because it allows us to get something that looks nice and allows us to control various things such as elevation, smoothness to give us something that looks realistic.**
2. **Grammars as we can select a lot of different tokens making each enemy/villager unique thus giving us a lot more control over the generation.**
3. **WFC as this generally takes the least amount of data, is pretty fast at generating content as we already have designer specs, we can generate content that easily follows these specifications.**
4. **Quest templates as these tend to follow a set guideline, these can be generated very quickly and can use things that have already been generated like location, NPCs etc. and use them to fill in the template.**

**Q7 (1 point)**. Rank the authorial burden (how much a designer would have to author/write/design) for the following techniques (numbered from 1-5, with 1 being the most burdensome) for the task of generating a variety of simple platformer levels: (a) Grammars, (b) Noise, (c) Genetic Algorithms, (d) Constraint Solver, and (E) Wave Function Collapse (**0.2 points** for each correct rank position). Assume you have a base, pseudocode-level implementation of any necessary algorithms to start, so all that’s left is the information specific to the simple platformer levels that the designer wants to generate.

1. **Constraint Solver**
2. **Grammar**
3. **Genetic Algorithm**
4. **Noise**
5. **WFC**

**Q8 (2.5 point).** Answer the below parts to the best of your ability.

**Q8.A. (1.5 points)** Consider your outputs from **Q5B** to be the random initial population for a genetic algorithm. Give the following: (1) a mutation function (**0.25 points**), (2) a crossover function (**0.5 points**), (3) a fitness function (**0.75 points**)that would give a 1.0 to a member of the population that fits all the constraints/rules from **Q5A** (i.e. acceptable numbers for each token type, path exists connecting player starting position, key, and locked door, and those three tokens aren’t neighbours). You may give your answer in pseudocode or plain text with sufficient detail that someone could write code based on what you’ve said. If you feel you could not give 4 unique outputs to **Q5B** then create 4 random outputs and give them here prior to your answers for 1-3.

1. **A mutation function could be given a mutation rate of 0.7 (0-1 assumption), swap the current element with the nearest neighbor starting from the left and then moving down, right, and then up.**
2. **A crossover function could be if there are two different non-player neighboring entities and merge their traits if the tokens remain valid. If there are two or more traits to an entity, take the first trait to merge. (i.e. making a locked door and a block to make a locked block).**
3. **If current entity is a player, check the surroundings and see if there are any valid neighbors which then returns one if all are valid and if the count is one. For keys and locked doors, we must check that the two are not neighboring each other and return one if that is true.**

**Q8.B. (0.5 points)** Roughly how many generations would it take for your initial population to converge given your answers to **Q8A**, if it would**?** Assume you are still using the outputs from **Q5B** as an initial population or the four examples you made for Q8A. Feel free to give a range. You do not have to show your work, but you can if you like.

**20-36**

**Q8.C. (0.5 points**). Given all of this, would you prefer to use WFC or genetic algorithms for this problem (generating dungeon “rooms”)? Give a brief justification (1-2 sentences).

**Genetic Algorithms as we are more likely to get a playable level which can include all different tokens than that of a WFC which has a chance to generate one which could not include all different tokens**

**Extra Credit (0.5 points).** In the video lectures we briefly discussed generating quests with a planner-based approach. At a high-level, describe how you would use an agent-based forward planner to generate personalized player quests. A perfect answer should include a pipeline for how you could go from data (with an example of a specific datapoint) about a particular player to new quests that, that the player would be more likely to enjoy. Make up whatever details you need to answer the question and make sure to state any assumptions.