1. (a) State in your own words the purpose and main tasks of a game engine. A game engine can share many parts and functionalities with a game. What separates a game from a game engine - what makes a game engine different from just a game (and vice versa)?

Answer: The purpose and the main tasks of a game engine is to provide the framework to build and create video games. Main tasks include rendering graphics, collision detection, memory management, animation and provide artificial intelligence or the game. Even though they share many functionalities, what separates them is that game engine is normally extensible to be used as the foundation for many different games, but a game normally contains hard-coded logic for special cases which makes it difficult to reuse or to make into a different game. Another thing that separates a game engine from a game is that a game consists of interactive experience that provides the player with an increasingly challenging sequence of patterns which he or she learns and eventually masters whereas a game engine is just providing a selected set of frameworks to build up a game.

(b) Find out what it means for a game or a game engine to be "data-driven". Give examples from your experience, or from available sources.

Answer: A data driven game/ game engine have the game contents and logic to be driven by external data sources, and it provides tools to allow non-programmers to do game design independently. Example of it is separating the game logic code and resources management code (like memory, network etc), so that the game logic code can be driven by external data sources, and we can make changes to it directly without affecting the resources management code (such as changing the gravity value of the text file).

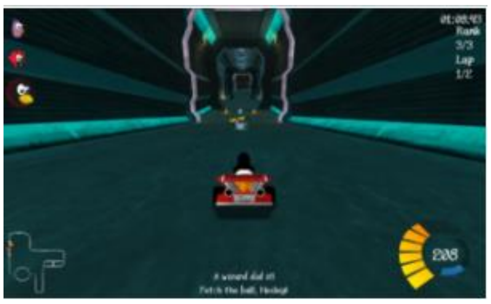
(c) What is the most popular programming language used for implementing game engines and their components? Try to explain why. What are its main characteristics (pros and cons) as an implementation language?

Answer: The most popular programming language used for implementing game engines and their components is C++. C++ is highly portable which is useful for developing games for multiplatform. C++ allows you to have total control over memory management which allows users to optimise the memory usage which is useful in heavy tasks such as rendering the graphic, however this can serve as a liability as well such as memory leaks. C++ has a large community support in which you would be able to get helps easily if you encounter any problems. C++ also has extensive libraries which come in handy for designing and implementing complex graphics, however, learning C++ can be difficult and keep people away.

2. Explain the following items concerning computer games and their development. Illustrate them with appropriate examples. 1. heads-up display (HUD) 2. third-person game 3. game mechanics 4. game platforms 5. game genre classifications 6. digital content creation tool 7. a particle system 8. asset conditioning pipeline.

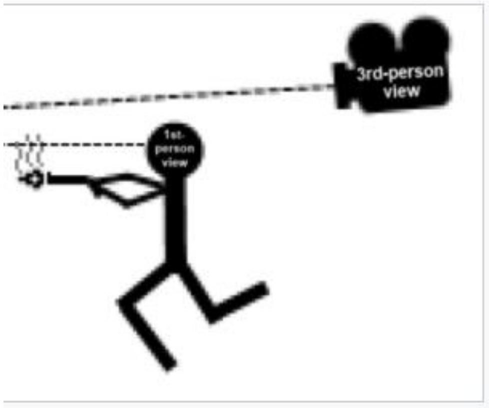
Answer:

1) Heads-up display (HUD) is the status bar which relays the information visually to the player as part of the game's interface.



The HUD displays the character's speed, rank, time elapsed, amount of fuel and also the map.

2) Third person game is game where the playable character is visible on-screen during play and we would be playing from third-person view.



3) Game mechanics are the rules that guide the player's moves or actions as well as the game's response to them, they specify how the game will work for the players.
4) Game platforms are the devices the users play the games on such as PC, consoles, mobile, VR, arcade etc.
5) Game genre classifications are the categories of games related by the similar gameplay characteristics such as first person shooter, platform games, fighting games, strategy games etc.
6) Digital content creation tool is a tool used to create digital contents that would be used in the games such as Cinema4D, Blender, After Effects etc.
7) A particle system is a collection of many small particles that can be used to represent a fuzzy object such as fire, smoke, snowstorm, fog, clouds, water etc in the game.
8) Asset conditioning pipeline is the workflow used to get the models, textures and sounds from your digital content creation tool by converting them to a file format that is most suitable for our game.

3. Explain the following items concerning C features:

a. How are primitive arrays and character strings implemented in C?

Answer:

Primitive arrays are fixed-size sequential collection of elements of the same type, it can be implemented in C this way.

To declare the array which can hold object with the arraySize of the type ‘primitivetype’:

primitivetype arrayName [arraySize];

To initialize arrays we can either

double balance[10] = {1000.0, 2.0, 3.4, 7.0, 50.0};

in which the number of values between the braces {} cannot be larger than the number of elements between the [] braces.

double balance[] = {1000.0, 2.0, 3.4, 7.0, 50.0};

or we can just initialize the array this way in which the size of the array is omitted and we will create the array of just enough size.

Character strings are defined as an array of characters in C and it is terminated with a null character ‘\0’.

It can be implemented in the following

char str[] = " hey ";

char str[50] = "hey";

char str[] = {'h','e','y','\0'};

char str[4] = {'h','e','y’,'\0'};

b. Assuming that the variables s and t are both pointers to characters, what does the following C statement do? Explain in detail how the processing proceeds: while (*s++ = *t++); (Such condensed code represents common idiomatic C usage even if possibly not best programming style.)

Answer: the content of t is copied to s character, then s++ and t++ would make s and t point to the next character, the while loop would ensure that it terminates when t has reached the null character 0.

4. Explain the following items concerning C features:

a. How does C manage (check, analyze, or prevent) uninitialized variables?

Answer:

Uninitialized variable has an undefined value, corresponding to the data that was already in the particular memory location that the variable is using, which can lead to errors that are hard to detect since the variable's value is random.

C does not have a built-in function to check whether if a variable has been initialized, but we can initialize the value to -1 if we know that the value would never be negative and test for if the variable <0 printf("variable not initialized") to check that if it has been initialized.

b. What does the built-in operation sizeof take as argument? What does it compute? Why is it a necessary feature in C?

Answer:

The sizeof operator can be used with any data types such as int, float, char… etc in which it returns the amount of memory allocated to that data types. It is a necessary feature in C as the sizes of data types may be defined differently for different platforms of implementation. So by using sizeof we can ensure that there is enough memory allocated for the variables we are holding.