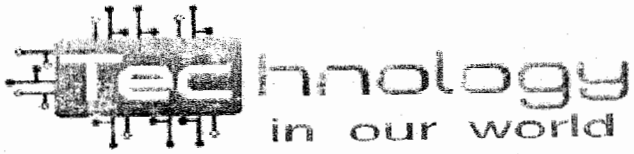
Name _____ Class_____ Computer_____ Date _____



Computer Game

How simply could we model a computer game? Let's start by separating the game in two parts: the data inside the computer, and everything outside the computer. This might seem like a strange thing to say, but remember that games are, first and foremost, interactive software. So, outside the computer, we have (at the very least) the *player.* Inside the computer we have all sorts of data, but the most important is what we call the *game state*: it contains all the dynamic information of the game: the position of the player, monsters and camera; the current time; the score – in essence, everything that changes with time.

A computer game can then be understood as a system in which those two entities – *player* and *game state* – interact with each other according to a specific pattern of rules. The player *inputs* data into the system, and the system uses the game state to generate *output* to display to the player. This is not very different from ordinary applications, in which e.g. a click (input) might cause a dialog to show up (output). The difference is how that behaves in respect with time.

Game structure

The central component of any game, from a programming standpoint, is the *game loop.* The game loop allows the game to run smoothly regardless of a user's input or lack thereof.

Most traditional software programs respond to user input and do nothing without it. For example, a word processor formats words and text as a user types. If the user doesn't type anything, the word processor does nothing. First, the game reads input from the user and stores it somewhere. Here, it will check the keyboard, gamepad, plastic guitar and network message queues – basically, it will collect all information from the outside world.

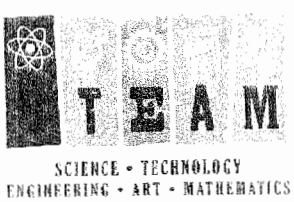
After that, it will use that information to update the game state – depending on what conditions we have, they will have different results. For example, on the menu screen, detecting that the "down" arrow got pressed might increment the "currently selected menu item" variable, but the same input while on the game might instead trigger the "set player as moving down" flag. Note that the word update is invoked even if the user didn't perform any input. In most cases the computer updates 60 times in one second. This is a very important property of most video games. This is called a loop. With games, code is constantly being invoked, even in the absence of user input, This makes the game look like it is always running or being played.

Lastly, it will collect the current game state data to generate the output – it will figure out where the player and camera and everything else are, and generate an image to display on the screen. A more complicated game will also have to deal with other forms of output – audio and network.

The vast majority of the game code will run inside the main loop. Game code is separated in three functions: processing input, updating the game state, and rendering the screen. Since all three of them are controlled by the loop, it's the foundation block of every game.

http://higherorderfun.com/blog/2010/08/17/understanding-the-game-main-loop/

fred
48



Name_____ Class_____ Computer_____

Parent Signature _____

What would you consider to be input in a computer game?
Explain as best you can, a Loop in a video game.
Explain the Game State

Standards RST 6-8.1, RI.6.1, RI 6.2

lexile level 980

48 fracl