

Structure Of Games

Sujoy Bit

Swami Vivekananda University

sujoybit3d@gmail.com

Abstract

Video Gaming as it is fun to play it is very addictive too. The game is created in such a way that the players are immersed completely in it. The game designers create the world in such a way that the players find themselves one with the games. Each of the elements is very methodologically designed to engross the players psyche and draw him or her in its world.

Key Words

User Interface, Main Menu, Game play, Influence, Game World, Main Loop, Time

Introduction

Video Games is more than just a word, it is a technology. They are perhaps the most powerful digital form of entertainment, which involves almost the entire attention of the players and draws him inside the game world. To achieve these video games are structured in such a way that not only they make things easy to create but also engage the gamers.

Components of Games

The User Interface- As described at the beginning, the user interface is made up of sprites, menus and so forth. It's what the user is given to control the actions within the game. These graphics are defined as buttons' which can be pushed, or a character which can be moved by the arrow keys. All of these elements are a part of the user interface.

The Main Menu- To start off with, just about every video game boots up to a main menu. This is usually a screen with some type of background, with an arrangement of buttons for

actions such as new game or starts game, options, load game and quit game. This screen acts as a control panel for the game, allowing the player to change settings, choose modes, or access the actual game.

Sometimes, a game will use the main menu as the in game menu. The in game menu is usually accessed by the escape key or the start button during game play. The in game menu allows the player to access most of the main menu actions with additional ones such as displaying character stats, points, inventory and so forth. Not all menus have to be squares with words in them though. The game “Secret of Mana” uses a creative menu where the level stays in focus while the choices form a circle about the player.

Starting the game

When you first start up the game a series of splash screens are shown. A splash screen contains elements such as logos, movies, and so forth. This often is used to tell the player the companies that contributed to the game firsthand and sometimes gives part or whole introduction to the plot. When the actual game has started there is often an introductory movie referred as opening cinematic that gives the prologue to the plot. This is not a movie like you see in the theater but usually a better rendered use of the game’s own graphics and sounds. In some games, you will then be asked for your name and in some games you will be allowed to customize your character, settings and so forth.

The next stage of the game is called the tutorial. It is not always considered a part of the game’s plot, but in some games it is integrated into the game that even though it is the tutorial stage it is a part of the game’s plot anyways. We will call this tutorial integration. It is widely used in games such as The Legend of Zelda and Super Mario 64. Few games give an option to either take the tutorial, e.g. Elden Ring, where the player can enter the tutorial area or by pass it to enter the game world.

Playing the game

During game play there are some basic concepts that just about every game uses. They are listed below:

Player-character relationship-The player's role in the character. How does the player control the character? There are usually 3 types of PCR, 3rd person, 1st person and influence:

3rd person: The player is not the character but instead is controlling the character/characters impersonally.

1st Person: The player is the character/characters - and sees things from the characters point of view both personally and visually.

Influence: The player is not tied to any character/characters but merely has an influence in the game. This is seen in puzzle games such as Tetris, and also in RTS games. There are some games where payer's decision and actions can change or influence the flow of the story as well as influence the ending of the story.

Game world

What is the 'world' that is portrayed by the game? It involves the story or the lore of the game which the player will explore throughout the game. There are 2 considerations in this-

Character role: There is also the question of what role the character plays in the game itself, there are 3 types in this sense.

Protagonist: Everything revolves around the character/characters the save the day type deal. Seen in games such as Zelda, Mario, Final Fantasy, and so forth.

Arcadic Conventional: An impersonal arcade character. Normally these characters are designed cartoony and don't resemble or follow any specific character personality or trait. Many NPCs are treated as impersonal characters, with which players can interact but do not make any everlasting mark.

Influence: Some games show a faceless influence in the game lore. They are not the traditional protagonist or antagonist. One such game STALKER- Heart of Chernobyl this element is very much highlighted. Apart from the main hero or villain the main influence in this game is the environment and the lore of the game, which very much influences the story,

character development and even the story ending. These types normally tend to follow a nonlinear pattern.

Law- Law refer to the general rules that governs a game, it also governs the game play as well. It dictates the progression, the pattern of the game, what to do and what not to do. They also provide nonlinear choices, which the player is free to decide and trigger different endings.

Graphics and Sound- Look and feel of the game. It determines how visually appealing the game is. Most of the AAA games has very high resolution graphics which makes the games a visual treat. The audio is also a major part of the game design; the sound enhances the overall atmosphere. Games like STALKER, Silent Hill, Dead Space deepens the involvement of the players.

Gameplay- This is the part which really engages the payer. This is the most important part of a game. A video game is not meant to be seen like a movie, it needs to be played and it is this involvement that draws a player in the game. No matter how much the game is visually attractive, if the gameplay is not engaging, then the player interest is lost.

Saving/Loading

Considering the saving and loading of the game, usually this can be a basic menu action wherein the player types a save name and the game is saved. In some games though, more creative approaches are taken so that the player is not pulled out of the gaming experience. Metroid Prime, Dead Space does this with its save stations. Loading however is usually a menu action.

The Main Loop

At the heart of our game is the main loop (or game loop). Like most interactive programs, our game runs until we tell it to stop. Each cycle through the loop is like the heart beat of the game. The main loop of a real time game is often tied to the video update (vsync). If our main loop is synchronized to a fixed time hardware event, such as a vsync, then we must keep the total processing time for each update call under that time interval or our game will “chug.” Each console manufacturer

has their own standards for video game publication, but most require that the game should provide visual feedback within the first few seconds of starting. As a general design guideline, it is desirable to provide the player with feedback as quickly as possible. For this reason most start up and shut down code is usually processed from within the main loop. Lengthy start up and shut down code can either run in a sub thread monitored from the main update () or sliced into small chunks and executed in order from within the update () routine itself.

State Machine

Even without considering the various modes of play within the game itself, most game code will belong to one of several states. A game might contain the following states and sub states: start up, licenses, introductory movie, front end, game options, sound options, video options, loading screen, main game, introduction, game play, game modes, pause options, end game movie, credits, shut down.

Time

A game loop must consider both how much real time has passed and how much game time has passed. Separating the two makes slow motion (i.e. Bullet Time) effects, pause states and debugging much easier. If you intend to make a game that can rewind time, like Blinx or Sands of Time you will need to be able to run the game loop forward while running the game time backwards.

Another consideration surrounding time depends on whether you want to go for a fixed or variable frame rate. Fixed frame rates can simplify much of the maths and timings within the game but they can make the game much harder to port internationally (e.g. going from 60 Hz TVs in the US to 50 Hz TVs in Europe.) For this reason it is advisable to pass frame time as a variable even if the value never changes. Fixed frame rates suffer from stuttering when the work load per frame reaches the limits and this can feel worse than a lower frame rate.

Variable frame rates, on the other hand, automatically compensate for different TV refresh rates. But variable rates often feel soggy in comparison to fixed rate games. Debugging, particularly debugging timing and physics issues, is usually more difficult with variable time. When implementing timing in your code there are often several hardware timers available on a given platform, often with different resolutions, overheads for accessing them and latencies. Pay special attention to the real time clocks available. You must use a clock with a high enough resolution, while not using excessive precision. You might need to handle the case where the clock wraps (for example, a 32-bit nanosecond timer will overflow back to zero every 2^{32} nanoseconds which is only 4.2949673 seconds).

Loading

Modern games are usually loaded either directly from CD or indirectly from the hard drive. Either way, your game could spend a significant amount of time in I/O access. Disc access, particularly CD and DVD access, is a lot slower than the rest of the game. Many console manufacturers make it a standard that all disc access must be indicated visually; and that is not a bad design choice anyway. However, most disc access API functions (particularly those that map through the standard I/O of the C runtime library) stall the processor until the transfer is complete. This is called synchronous access.

Multi-threaded disc access

One way to get feedback while accessing the disc is to run disc operations in their own thread. This has the advantage of allowing other processing to continue, including drawing some visual feedback of the disc operation. But the cost is that there is a lot more code to write and access to resources needs to be synchronized.

Asynchronous disc access

Some console operating system API's handle some of the multi-threading code for you by allowing disc access to be scheduled with asynchronous read operations. Asynchronous reads can tell that they are done either by polling with the file handle or using a callback.

Renderable Objects

Whether a game uses 2D graphics, 3D graphics, or a combination of both, the engine should handle them similarly. There are three main things to consider. Certain objects may take a while to load, and can momentarily freeze the game. Some machines run slower than others, and the game play must continue with a low frame rate. Some machines run faster, and animation could be smoother than the time interval with a higher frame rate.

Therefore, it is a good idea to create a base class as an interface that separates these functions. This way, every draw able object can be treated the same way, all loading can be done at the same time (for load screens), and all drawing can be done independently of the time interval. OpenGL also requires object display lists to have a unique integer identifier, so we'll also need support for assigning that value. One common method of collision detecting is by using axis-aligned bounding boxes. To implement this, we will build upon our previous interface, `IDrawable`. It should remain separate from `IDrawable`, because after all, not every object drawn on the screen will require collision detecting. A 3D box should be defined by six values: *x*, *y*, *z*, width, height, and depth. The box should also return the object's current minimum and maximum values in space.

Conclusion

Video games, fun to play and they looks pretty interesting and gorgeous; but what we seldom notice the intricacies and the components and structures required creating such visual masterpieces. No matter how much the game experiences differ or how different the game genres may be these basic structure and components are applicable.

Reference

- Story and Narrative Structures in Computer Games-Craig A. Lindley
- Game Design Workshop – Tracy Fullerton

The Art of Game Design: A Book of Lenses by Jesse Schell.

Classic Game Design by Franz Lanzinger

Advanced Game Design by Michael Sellers