COMP 369 – Assignment 3 Part 1

Mathew Boland – 3359644

Research and Report

Introduction

In this research and report I will be discussing and explaining the various different types of backgrounds used by and implemented in various video games. For each of the differing styles I will go over what added benefits they have in terms of gameplay, scenery and overall effect on game quality. As well as with the good, I will discuss the negatives of each implementation for those same aspects. Afterwards I will discuss what technology is needed to implement such a background along with a couple of examples of each in practice. The goal of this research is to shine a light on what makes each background implementation worth it over the others depending on what game is aimed at being developed.

Solid/Image Backgrounds

The Good:

With a solid background implementation is easier than with other types of video game backgrounds. Due to the simplicity of making the background a single set colour or of a static image, much less complexity is needed in running and designing the game. An added benefit as well is that it can make testing easier during development. Having a plain background or rather any background that isn't distracting can make testing and checking sprite functionality an easier process.

There is really only one other benefit to this, other than that there is at least some sort of background with solid backgrounds. That benefit is that it can improve the performance of a game. Back in the earlier retro era this could be quite significant. Still, having at least a static image or textured background was much more common as it added to the quality of the game significantly without reducing performance much. This being that a solid background was still used, just one that consumed larger memory but not processing power.

The Bad:

It is much uglier. The alternatives just offer a much more vibrant and significantly improved visual experience. Static solid images could still be interesting but pale in comparison to the liveliness of an animated or scrolling background.

The improved performance is by all means negligible now a days. The extra computing needed to improve backgrounds beyond this is something that would have been somewhat significant back 30 years ago and would have meant maybe removing a level or some aspect of gameplay to accommodate the hardware otherwise. Now though, essentially all hardware is orders of magnitude more powerful than back then and the performance impact is non-existent. That is, at least with the basic implementations of the following backgrounds, they can be taken to extremes that would even make a modern day computer chug like any software could.

Technology:

Generally not a lot of technology is needed to implement this kind of background in a game. Solid colour, depending on what engine is used, can be implemented by drawing a single colour over the entire game area before anything else is drawn. Images can be more complex but usually just rely on needing some sort of technology capable of displaying an image. Then that image is drawn before anything in the foreground much like the solid colour.

Examples:

Old sports games such as the ones for the NES and earlier generally only used single colours for the background of their games. This allowed for the needed backdrop of usually a green field to be implemented easily and with little cost to the performance while still providing the necessary look for the game.

Other games include ones such as Tetris or Snake. These games were quite primitive and relied solely on their gameplay antics being drawn over a static background. It did not matter what the background was for these games since it did not impact the gameplay. Additionally, more detailed backgrounds could have been distracting and detrimental to the players experience.

Animated Backgrounds

The Good:

These backgrounds provide a much more vibrant and interesting back drop for a game that usually makes the scenery more interesting while playing. Having a background that moves makes the game feel much more alive than one that is static. Seeing leaves fall from a tree or blow in the wind provides a much more immersive experience than that of a static tree image.

Backgrounds can be made interactable. More advanced animated backgrounds can be programmed to react to certain events in game and even to make it seem like things in the game are being affected by what is going on in the background. Using this, the lines between background and game area are blurred and the player can have a greater sense of freedom in experiencing the game.

The Bad:

The memory needed can be quite demanding, especially for the more ambitious backgrounds that are animated. Having to keep an animated GIF in memory occupies more space than a plain static image of the same quality due to the added frames. If backgrounds are implemented as sprites then this issue is exacerbated further. Animating a background like a sprite requires even more memory to be used and processing power handling the image coordinates and frame swapping.

It can also increase the time needed for development. Sometimes your basic animated background can be as simple as adding a static image background when things such as GIFs are used since many libraries support their use similarly. However if a sprite is used then additional functionality must be implemented into the program to account for changing the frames and drawing the background properly. Needless to say if a background is also interactive. Then it will also require added functions to handle these interactions, further adding to the amount of development needed.

Technology:

Technology is needed to render the background much like with a sprite. Usually the same or a similar system can be used for drawing an animated background. Animated backgrounds can be just like a sprite in that they have frames that are updated and rendered. However they could also just be a GIF that plays. More in depth backgrounds can be made to react to actions and events unfolding in the game as well by being programmed to do so.

Examples:

Pinball is an example of interactive animated backgrounds. The ball moves around the background, interacting and setting off events that change the background when needed. The animation allows for the needed feedback to the player on their performance in game.

Another would be fighting games. Games like Mortal Kombat use animated backgrounds to provide interactivity and immersion into the experience. Having chains sway in the prison backgrounds or even being able to interact with them by swinging from/jumping off and above an opponent using background objects adds to the quality of the gameplay.

Scrolling Backgrounds

The Good:

Scrolling backgrounds allow for the player to move their character in a manner that appears as though the game screen (or camera) is fixated on the playable character. This functionality allows for much more explorable game level design. Being allowed to move up, down, left or right to further explore the background makes a world feel more open and player controlled.

This design also allows for change in scenery. Progression along a level can lead the player further along the background and eventually to a different area of the background in the level. Allowing for a change of setting and/or tone. As well if this is not desired, then looping the background can be done in order to make the background constantly feel like its changing when in reality its just being recycled.

These types of backgrounds can also take advantage of the previous ones. Static images can be used with a zoomed camera (single colour wouldn't work/make sense with this though). Animated and even reactive backgrounds can be used as well, though this can and usually is much more difficult to do with this design as it further adds to the complexity.

The Bad:

Scrolling is a difficult and demanding feature to implement. When moving the player, the game must handle moving the background to keep up and stay with the player smoothly and seamlessly. Additionally, all sprites must be updated in this manner as well. The result is that the game must be designed to encapsulate the game screen minus the background so that it can stay stationary while the background moves around it which adds an extra layer to the complexity in designing a game.

Design needs to also take into consideration the need to wrap the background. If the game isn't designed to use one set image that fully encapsulates the playable area, then wrapping is needed. This will stop players from reaching an edge and ruining the experience. In order to do this, the background must be designed so that when it is wrapped around it does not have seams or anything that would make it feel like the background has split when restarting. As well, some type of technology must be used to do the wrapping so that one image can be used instead of the same one multiple times, hogging extra memory.

Technology:

In order to use scrolling backgrounds, additional code is needed to handle the view and game area in front of it. Programmers can handle this issue themselves or take advantage of many library features available in a lot of game engine APIs today. What these engines usually offer is what is called a camera and/or view. Developers can specify the game area, set a background and then create a view inside of that game to use. From then on that view is used as a basis for the player to see the game. Depending on what engine is used and what is being added, the coordinates needed to use for development will need to draw from that view or the entire game area. An example technology that uses this would be the Phaser 3 JavaScript game engine and its ability to handle multiple scenes/views and cameras for 2D games.

Examples:

Since this is the most common type implemented in well known games, it is easy to find many examples. Good ones would include the original Super Mario Brothers on NES (not to be confused with the original Mario Bros game for arcade and Game and Watch that used static backgrounds) as well as every following main Mario game until Mario 64. These games would all feature a camera focused on Mario with the background moving along as the player moved, usually faster than the foreground.

Many other examples include: Mega Man, Sonic, Castlevania and Metroid to name a few. All of which took advantage of scrolling backgrounds that moved along with the player. Some of these would only move from left to right. Others would move in every direction such as the ones used in the Metroid franchise.

Closing Remarks

Thinking about the majority of 2D games that are/have been made, it is no wonder the majority go with scrolling backgrounds as their design choice. Scrolling backgrounds offer the most unique and custom gameplay opportunity for exploring levels and creating unique environments. That is not to say that all games should use it though. If a games gameplay mechanics lead there to be no use of expanding a level beyond its initial view such as in Tetris or Pinball. Then it is better to choose animated backgrounds if the background will still play a major role in the gameplay and to choose a solid or static image background if it will not. For virtually all other games though, scrolling backgrounds are needed so that the game world can expand throughout a level and not stay confined to its initial view. Still, each type can be more beneficial depending on the scenario of the game and it is always important to consider which one is best suited for a game based on its mechanics before diving into development.

Works Cited

- Morrison, Michael. “Understanding the Types of Game Backgrounds.” *Understanding the Types of Game Backgrounds | Sams Teach Yourself Game Programming in 24 Hours*, 2002, flylib.com/books/en/4.43.1.160/1/.