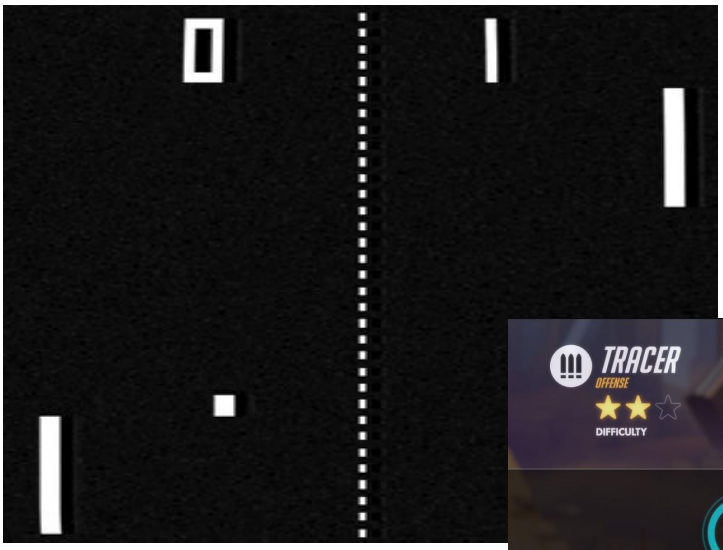
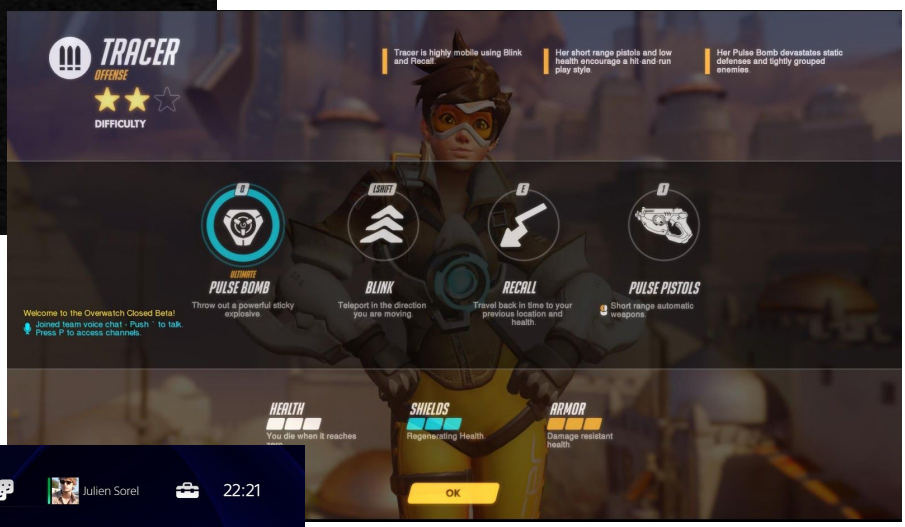
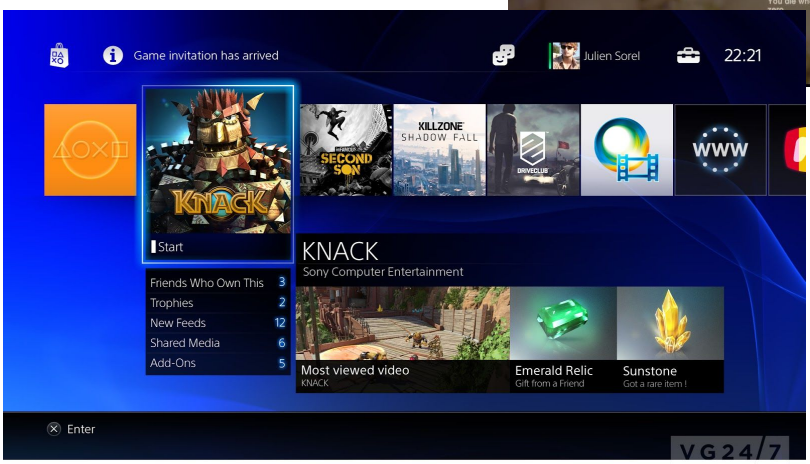
UI Plan







Roger Busquets Duran

Index

Introduction

User Interface

The term user interface is not only applied in video games. This term refers to the parts of the software that interact directly with a human. It is the responsible of communicating with the player, telling him which options are available at any moment. So the UI is mostly related with design. In video games user interfaces are generally divided in two parts:

- Input: How the player gives commands and communicate with the system.
- Output: How the game communicates the results of those actions and other aspects of the game state to the player.

A user interface is present in all kind of software, but we can also consider other types of UI. For example, what about a non-digital game? In non-digital games, the UI is the game's components themselves, since they are both how you interact with the game (through manipulating the game bits) and receive feedback (by viewing the game state).

League of Legend user interface.

UI Plan

Now that we know what is a user interface lets focus on how to plan it.

Since we are talking about planning the UI first we will talk about designing it, and then how to develop it.

UI Design

First of all let's define which are the facts that makes a UI good.

- **Ease of use**: How fast and easy is for the player to perform the action or task correctly.
- **Ease of learning:** How easy is to figure out, for those who are new in the game, what the player is allowed to do and which is the information available.

Depending for which platform your game is going to be developed you can make different decisions to achieve those previous aspects. For example for a PC game the presence of hotkeys can ease the use, since it is always faster to press a couple of keys for the player.

When designing a UI we have to have clear to concepts:

- **User model**: The user model is how the user (in our case the player) thinks that things should work.
- **Program model**: The program model is how the software actually works. The program model is always right.

The perfect situation is that both models match perfectly, otherwise the player will expect a different output, will get frustrated and will stop playing the game. So our goal is to design our UI so the program model fits perfectly with the player expectations. How to do it? **Playtest!**

The best way to check if what we are trying to communicate through the UI is clear for the player is testing it with real players.

To sum up a good UI should accomplish the following aspects:

- It does what the user thinks it will do.
- It gives the user feedback so they *know* what it did.

As you go about designing the UI, here is a process you can follow:

- First, make a list of tasks that players need to accomplish in the game. Make it easy to do those tasks.
- Second, pay special attention to the most *common* tasks, the ones that players are doing over and over. Difficulty and complexity of a task should be in proportion to how rarely it is used.
- Iterate through playtesting.

One planning tool for designing the UI is a wireframe (a blueprint that shows the arrangement of content on the screen). A wireframe in your design document will help you and your team members to create a good UI.

When designing the UI for your games it will be helpful to ask yourself the following questions:

- Does this interface tell me what I need to know *right now*?
- Is it easy to find the information I'm looking for, or do I have to look around for it? (Are the menus nested so deep that they hide information from the player?)
- Can I use this interface without having to read instructions elsewhere?
- Are the things I can do on this screen obvious?
- Do I ever need to wait for the interface to load or play an animation?
- Are there any tedious or repetitive tasks that I can shorten (with a shortcut key, for example) or remove entirely?

Use of Color

Color is a powerful tool to communicate with the player, it catches easily the player's attention and it takes up no additional space on a game component.

Red, green and blue are the colors that catches easily player's attention so they may be used in important components of the UI. Colors also have a meaning, for example red will express alert, this is an international code, so it is not a good idea to use the red color for giving good news to the player.

Obviously do not overuse the same colors, vary the intensity of colors as well , and use different shapes in addition to different colors. By having things differentiated in multiple ways (different shape, different color, etc.) it makes it really obvious that those things are distinct from each other.

Use color consistently. If you use the same color for several things in your game they may be related, so be careful when choosing a color for a new component since players can easily be confused and think that that component has a relation with another one (this will be the situation that we want to avoid, user model not matching with the program model).

Diegesis Theory

User interface in games differs from UI design since in video games we have an important factor that do not appear in other software applications: **FICTION**

This fiction can be directly linked to the UI, partially linked or not linked at all.
There are several theories that can be used by designers to analyse a user interface and help them break down choices. The theory we will look at here is called *diegesis theory*. It is adapted from diegesis theory used in literature, film and theatre. Diegesis refers to the world in which the story is set, and hence it focuses on games as stories.

There are two concepts core to this theory: **narrative** and the **fourth wall.**

- **Narrative:** Narrative is a message that conveys the particulars of an act or occurrence or course of events. In simple terms, it is the story the designer wishes to convey. For example the story of blocks falling from the sky which need to land in the right place (*Tetris*). Not all elements of a game are part of the narration. For example, the game menus and the HUD, because the game's characters are not aware of these elements. This does not mean these components do not support the narrative. For example, a futuristic game typically has GUI elements that also appear futuristic.
- **Fourth Wall:** The fourth wall is the imaginary divide between the player and the world of the game. In order for the player to immerse themselves in the game world, he needs to move through the fourth wall. The ease with which the player moves between the real world and the game world depends on the way the interface designer delivers information to the player.

Taking in account the relation of the UI with the fiction of the game we can ask ourselves the following questions to classify our UI elements.

- Is the component part of the game story? (Is it part of the narrative?)
- Is the component part of the game space? (Is it behind the fourth wall?)

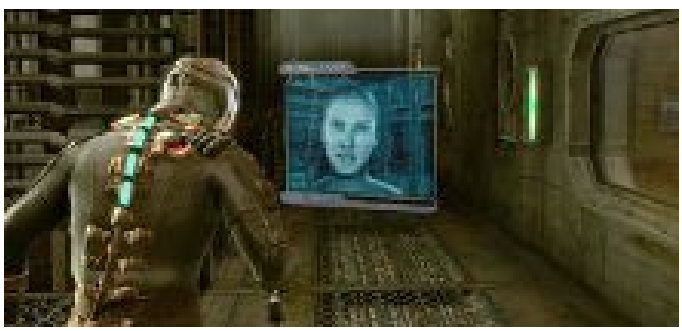
Diegetic Components

Is it part of the narrative? **Yes**
Is it behind the fourth wall? **Yes**

Diegetic components provide the player with cues and information without distracting them from the narration of the world. These cues are something that the player's avatar and other characters in the game world are aware of, and can interact with. Diegetic user interface elements exist within the game world (fiction and geometry). Well executed diegetic UI elements enhance the narrative experience for the player, providing a more immersive and integrated experience.

In *Far Cry 2* there is no HUD, so the player receives information through numerous in-game widgets and items without referring to outside elements or overimposed elements in the game reality.

Metro 2033 uses a complete Diegetic UI with no HUD elements to help to support the game's narrative. It runs the risk of frustrating the player though slow response times but this forms part of the game mechanic. The character's watch is used to measure how long the filter in the gas mask will last and how visible he is.



In *Dead Space*, instead of providing a typical health bar overlay, the player's health is indicated by the high-tech meter on the avatar's suit.

In real time strategy games,for example, diegetic components are elements such as the visual damage to units and buildings.

This kind of UI is often used in futuristic games where UI overlays in daily life are commonly accepted.

Designing diegetic interface components to replace common HUD elements requires a clever approach. While this is great for the immersion of the game, if it is not done correctly, it can have the opposite effect.

In the adventure game *Grim Fandango* the player is forced to search through their inventory one item at a time. This frustrating process breaks the player's suspension of disbelief, and he pops back into reality.

Depending on the type of game you are designing, a completely immersive and realistic world may not be what you are looking to achieve, and this may in fact break the narrative of your story.

Non-Diegetic Components

Is it part of the narrative? **No**
Is it behind the fourth wall? **No**

These elements have the freedom to be completely removed from the game's fiction and geometry and can adopt their own visual treatment, though often influenced by the game's art direction. This elements are used when diegetic elements could distract too much the player or break the consistency or legibly of the UI element. Non-diegetic components allows the player to easily customize the UI so it gives a more familiar experience.

Most of the UI elements in World of Warcraft sit on the 2D hub plane

Smite non-diegetic UI

Some elements can be confused while classifying them as non-diegetic for example the speedometer in the HUD of a racing game.

Spatial Components

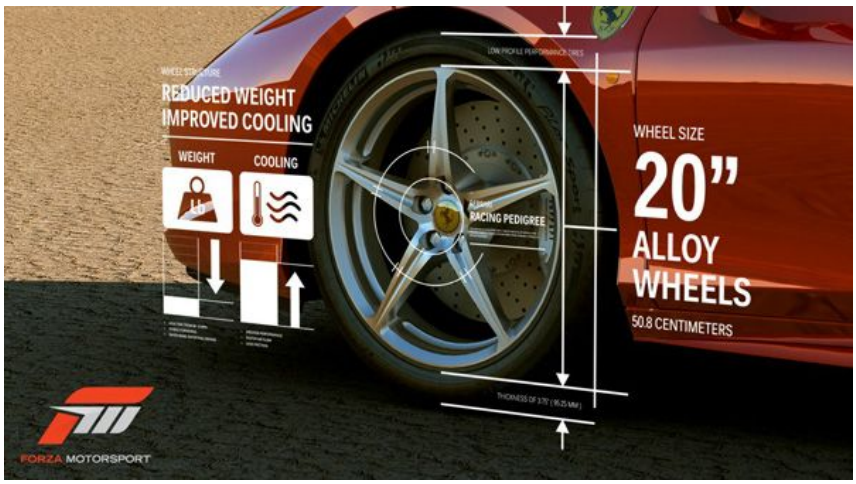
Is it part of the narrative? **No**
Is it behind the fourth wall? **Yes**

These are components that are visualised within the game world but are not part of the game world. The game's characters are also unaware of these spatial components. For example, the aura selection brackets around units in real time strategy games. They are used to provide extra information on a component in the world, although that information is not part of the narrative. The information is provided in the location on which the player is focused, reducing clutter in the HUD.

Spatial UI elements are used when there's a need to break the narrative in order to provide more information to the player than the character should be aware of. They still sit within the geometry of the game's environment to help immerse the player and prevent them from having to break the experience by jumping to menu screens. The closer these follow the rules of the game's fiction the more they can help immerse the player.

Splinter Cell Conviction

Fable 3



Forza 4

Spatial elements can be beautiful pieces when they work with the geometry of the world.

Meta Components

Is it part of the narrative? **Yes**
Is it behind the fourth wall? **No**

Meta representations are components that are expressed as part of the narrative, but not as part the game world. These can be effects that are rendered onto the screen such as cracked glass and blood splatters – effects that interact with the fourth wall are the most common examples.

These components aim to draw the user into the reality of the game by applying cues to the screen as if the game were directly interacting with the player.

Call of Duty: Modern Warfare 2.

Blood splashing

Interacting with the phone in *Grand Theft Auto 4*

As said before the speedometer can be confusing when classifying it, but I would consider it a meta component since I think it is part of the narrative, the character driving the car can see the speedometer, so is a representation of what the character can see.

Diegesis theory in 2D games

Most of the examples that has been shown are from 3D games, but it does not mean that this theory cannot be applied in 2D games. A 2D world is a flattened representation of a 3D world. So in a 2D game we still have narrative and the fourth wall, then all this theory can be applied equally.

Taking Pac-man as an example, we can identify the concept of a world or narrative component and components that are outside this world. Pills, walls and ghosts would be all diegetic components, the ghosts changing color when can be eaten would be a spatial component, and the scores and details around the game would be non-diegetic.

Development

Once we have very clear the design and how our UI will look like is time to develop, and implement it in the game.

Tools

- Pen and paper (sketching, used during the design process)
- Photoshop(or other illustration programs): To create the sprites that will be used as UI elements.
- Axure(or other UI prototyping programs): can be useful to how the UI will look like in-game (not implemented yet)
- External libraries that allows to create the code necessary to display UI elements in the game.
- C++, patience, and develop your own UI system.

Art

First we need the UI atlas, where we can find all the sprites needed to reproduce the design of our UI. In huge studios there may be a UI artist and a UI designer, but it can perfectly be done by the same person. Anyway the responsible of creating this atlas will have to produce something like this:

A well organized spritesheet with all windows, buttons, labels, icons… needed. The easiest way to do that is using photoshop, create all those elements separately and finally put them all together in the spritesheet. The person who is going to make this sprites has to have clear how they have to look like, so it is important to have a design document that explains everything that the artist need to make the UI elements.

Code

Once we have the UI atlas ready the next step is to implement it in the game. This task requires coding skills. In video game studios there is the UI programmer who will be in charge of implementing the design of the UI in the game, so he will have a close relation with the UI designer and artist.

The coder has the option of creating a UI system from scratch or use external libraries that allows to create UI elements. This are a few examples of c++ UI external libraries:

- Qt: Qt is a full development framework with tools designed to streamline the creation of applications and user interfaces for desktop, embedded, and mobile platforms. https://www.qt.io/developers/ http://doc.qt.io/qt-5/
- Adam & Eve from the Adobe Source Library (ASL): It relies heavily on the Boost libraries(provides free peer-reviewed portable C++ source libraries). https://stlab.adobe.com/group__asl__overview.html
- WTL (windows template library):Windows Template Library (WTL) is a C++ library for developing Windows applications and UI components. It extends ATL (Active Template Library) and provides a set of classes for controls, dialogs, frame windows, GDI objects, and more. http://wtl.sourceforge.net/

If the coder decides to develop a UI system by himself a good starting point would be having clear which UI elements should it be able to reproduce (image, button, label, text input, window, minimap...), and then make a UML with the basic structure of the code. Having the idea clear before start coding.

Planning

Now that we know the basics of UI design and the goals that we as designers should be accomplishing in order to make a good UI, let's focus on which would be the workflow. Of course there's more than one way of doing it, this is my proposal.
I would divide the development process in four phases:

- **Phase 1: Research and Requirements:** Make a research about which are the trends at the moment, what do players expect, that means interviewing players, analyzing other successful game's user interfaces and with the information gathered and the theory learned about UI design start creating the first designs and documentation of our UI.
- **Phase 2: Concept Discovery:** Have a final design of the UI, and the final documentation of the UI plan in order to proceed with the development, after making some tests and making sure it is a valid UI.
- **Phase 3: Concept Development:** Have the art ready and a clear idea of how it is going to be implemented in the game. Then, start the development process of the UI,making also tests during the process, and making changes on it if necessary.
- **Phase 4: Test:** Once the UI is finished it is time to make a lot of tests, if possible, external playtests, to find usability issues and have feedback from our target, and be able to modify the UI. This way we make sure that our game is engaging, immersive and our UI do not disturb the player or interfere too much with the gameplay. So in result we get a pleasant UX (user experience). This can be a very expensive process that not all companies can afford.

Everything explained during the document is useful for preparing a UI plan document. This document should contain a design guide that explains how the UI should look like, how it would be displayed in the screen, and it can also contain how it is going to be implemented, from the technical point of view. So it is a mix between design, art and code.

Interesting Links

GDC speech of the UI designer of *Hearthstone* UI about how to make an immersive user interface:

https://www.youtube.com/watch?v=axkPXCNjOh8

Interview to a UI design and artist (how to become a UI artist):

http://www.gameindustrycareerguide.com/how-to-become-a-video-game-ui-artist/

Documentation Links

https://gamedesignconcepts.wordpress.com/2009/08/24/level-17-user-interfaces/

https://gamedevelopment.tutsplus.com/tutorials/game-ui-by-example-a-crash-course-in-the-good-and-the-bad--gamedev-3943

https://en.wikibooks.org/wiki/Video_Game_Design/Chapters/Implementation#The_User_Interface

https://books.google.es/books?id=6g8NDgAAQBAJ&pg=PA179&lpg=PA179&dq=planning+UI+for+a+video+game&source=bl&ots=Zk9tQPeWmc&sig=8stNAx_Fib-EtpLiL5imaBtjJYY&hl=es&sa=X&ved=0ahUKEwjFhp7XiJ_SAhVEWBoKHYBQBbo4ChDoAQhHMAY#v=onepage&q=planning%20UI%20for%20a%20video%20game&f=false

http://www.gamasutra.com/blogs/AnthonyStonehouse/20140227/211823/User_interface_design_in_video_games.php

http://devmag.org.za/2011/02/02/video-game-user-interface-design-diegesis-theory/

https://www.quora.com/What-are-the-best-practices-for-UI-in-video-games

http://whatusersdo.com/blog/uxui-design-video-games/

http://www.gamasutra.com/view/feature/4286/game_ui_discoveries_what_players_.php?print=1

http://www.hongkiat.com/blog/video-games-ui-evolution/

https://wiki.duraspace.org/display/DEV/Project+Planning+the+User+Interface