**Objects in Video Games**

Within the overarching field of Software Studies, video games position themselves in a peculiar situation. On one hand, video games are clearly software. They consist of code, they are programmed by software developers, and they have many similarities to general software artefacts – both from an engineering perspective, but also from a critical and analytical perspective. On the other hand, video games are clearly also a matter for literary and narratological perspectives. Video games are essentially experienced as narratives. There is often a constructed narrative, a “story”, tagging along throughout the player’s progression of a video game, but generally the player’s interaction, and therefore the playing in general, constitutes the majority of the video game narrative. Scores, rules and systems indicate starting points, ending points, and all the progression in between. This progression is brought by the game itself, but more often than not the underlying software is dependent on the interaction of the player in order to initialize any progression. The narrative becomes an interactive narrative. Furthermore, video games also communicate themselves through audio-visual outputs - the notable example here being the GUI (graphical user interface). This means that the mediation of any underlying software and code is obscured to a certain degree, and usually not at all spoken about or explicitly addressed in the game narrative. Instead, the GUI mediates a very abstracted representation of the code that constitute the game. This is what the player immerses him- or herself into.  
Therefore, the study of video games has often addressed video games from this external appearance. Some scholars call this “screen essentialism” in terms of New Media in general, and through it a lot of the output of video games – sensed through interaction with the designed interfaces – is certainly uncovered. However, the actual underlying engineered software, the critical understandings of software, and more importantly for my paper, its relation to this given output, is still hidden. This is where Software Studies enter the picture.

*Expressive Processing: Digital Fictions, Computer Games and Software Studies* will probably be a book I refer to a lot throughout this paper. The author, Noah Wardrip-Fruin, argues that the underlying computational processes are increasingly authored in video games, and they are arguably then expressive in the video game output. He writes: “When I play a computer role-playing game (RPG), author-crafted processes determine how I can speak with the non-player characters (NPCs).” Within my text however, I will seek to not so much talk about the narrative implications of computational processes and therefore continue his author-focused perspective. Instead, I will offer a case oriented and speculative text that seeks to place *object-orientation*, or rather the abstraction of objects or entities from a programming concern, in relation to the video game presented in the output or the external appearance. It is my belief (and arguably also assumption hence the speculative aspect of the text) that such a programming structure has an incredibly influential role in both how games can be envisioned to be constructed through code, but also in how video games are able to translate often real-life mechanics into rule-based logic systems. These systems are constructed by people, and they depend on factors (e.g. entertainment value) very foreign to the real-life variation outside of the software.

In writing about object-orientation, I may or may not switch it to just objects instead. I still have to explore and understand how video games are programmed, as I have come to understand that object-oriented programming is often a bad idea for so complex structures such as contemporary video games. Instead, the development delves into such terms such as “entities” and “composition”. In the end, what I want to focus on is not so much the programming technique with its syntax, but instead its abstractive qualities and properties, and how they are expressed in the output of the game – how an object is defined by set arguments and parameters, how the object interacts with both player and system in the design of the game, and also how using such abstractions make for a certain type of game.

This “certain type of game” is one build from multiple objects that exist and interact with one another. In this game, the designers and developers are forced to put in many, many hours of work in order to properly define their set of rules and logics, and the overall narrative presentation of said rules and logics, in such a way so as to attain the player’s interest in the game and the equally important suspension of disbelief, and not alienate or bore the player. In order to present both a proper narrative and playing field, the developers have to consider how many rules and logics are sufficient, necessary or too many. A cluttering of mechanics can easily ruin a game, and besides the developers also have deadlines and budgets to consider.

Let me summarize. I wish to explore the topic of video games as seen from a software critical perspective. I wish to do this with a specific focus towards objects and the abstraction of values and parameters. Such a text can easily constitute an entire book, and thus I will delve into the topic using case examples, and from them, deriving at speculations, ideas and concerns that I believe can be taken up properly by new media- and software studies communities. I will begin my paper with an overall outline of objects and classes, the meaning of OOP, and also how such abstractions are used in video game development. Ultimately, if I do not find any answer for the very specific technicalities of video game development, I will still argue that the “*expressive processes*” found in the output of video games are sufficient enough to make me argue for object/entity-abstractions found in my case examples – even if I have not studied the source code (intellectual property).

I have not yet chosen my case examples. However, I can already see numerous approaches. Nevertheless, I hesitate in choosing specific ones as I am unsure if the amount of writing material will end up being more than fifteen pages. Here are a few examples. 1) In RPGs you have “stats”, which directly translates into the arguments and methods of an object, and you explicitly use those defined parameters in order to suitably progress the game. 2) Games also often have predefined objects as enemies. You are often supposed to learn the repetitive patterns of objects/enemies in order to surpass them. Interestingly, game design is then seen to use the same objects in new constellations and level designs in order to pose a different challenge. 3) In various video games you can often choose between embodying either a female or a male character. However, how does a game distinguish between male and female in relation to playable mechanics? More often than not, the difference is small, and constitutes often only certain narrative elements.

In terms of next steps, I will be looking for more literature. I need to find a focus for a last section of the paper wrapping up my text – I still need to make up my mind about the larger dilemmas at play here. I also need to be able to specify my boundaries in terms of concluding anything at all. As I said earlier, this will be a speculative text (not a lot of literature produced). I am considering using Berry’s *Philosophy of Software* to talk about translating the realness of the world into computational processes. I could also apply an interaction design perspective from *Things That Keep Us Busy* (I happen to have read it) in order to discuss the boundaries/possibilities for player interaction with objects of limited *interactability* (the ability of the artefact to be interacted with in different ways).