The evolution of FPS games controllers: how use progressively shaped their present design

Kostas GKIKAS† Dimitris NATHANAEL† Nicolas MARMARAS‡
dpsd00082@aegean.gr nathand@aegean.gr marmaras@central.ntua.gr

†University of the Aegean – Department of product and systems design engineering
‡National Technical University of Athens – ErgoU

Ermoupolis, GR-84100, Greece

Abstract

First Person Shooter (FPS) is a genre of computer games with immense success over the last decade. The genre has evolved rapidly thanks to technological advances in 3D graphics and Artificial Intelligence to high levels of realism. However, its hardware interface has received, or so it seems, little attention; most expert players finding a conventional keyboard and mouse configuration as the best choice. In this paper we take a close look at the evolution of FPS controller configuration. We try to show that this historically evolved control configuration, although not optimum in any analytical sense, blends in an original way diverse and often conflicting ergonomic requirements. We conclude by suggesting that the sustainability of the keyboard-mouse as the FPS controller of choice is mainly due to its inherent plasticity. This plasticity has facilitated not only the continuous streamlining of the controller configuration per-se but has also significantly influenced the genre's evolution as a whole.

Keywords: FPS games, controllers, mapping, task-artefact cycle

1. Introduction

The aim of human-computer interface design in general is to achieve high levels of usability. Usability is defined in ISO 9241-11 as "the extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use" [ISO (1998)]. However, play as an object of a human activity cannot be defined as accomplishment of a pre-specified set of goals [Koster (2004)]. Therefore nor efficiency, neither effectiveness, can be considered as critical dimensions in game interface design. In fact, the very concept of usability does not make much sense, at least in its common ISO 9241-11definition. Play, rather than task accomplishment can better be conceived as a balance between challenge and satisfaction. In play, contrary to work, if challenge levels (i.e. difficulty) fall below a certain level, satisfaction decreases as well. Thus, since players progressively become

more competent, games need to provide more and more challenges to keep of the same level of satisfaction [Gee (2003)].

The need for continuous increase in the challenge levels is considered explicitly in the game industry. It is usually achieved (i) by allowing for different levels of play in the same game and (ii) by re-designing challenges in new versions of the game's software. Common wisdom then suggests that games evolve by progressively altering their software to provide higher level and more diversified challenges. The hardware part of the games (i.e. the controller/s) is often not considered as having an important impact on this evolution. However, studying the evolution of FPS games we suggest a different picture.

2. First Person Shooter Games

First-person shooter (FPS) is a genre of computer and video games which is characterized by an on-screen view that simulates the in-game character's point of view and a focus on the use of handheld ranged weapons. They provide immersive, engaging, and highly interactive worlds that allow players to engage in behaviours similar to those in the real world.

The modern FPS genre emerged at the point when home computers became sufficiently powerful to draw basic 3D graphics in real time. id Software's® Wolfenstein 3D® and Doom® are widely considered to be the breakthrough games of the genre. The latter, in particular, defined the genre so emphatically that FPS games were commonly referred to as "Doom clones" or "Doom-likes" for a significant period after its release. Other notable examples of the genre include Quake®, Unreal®, the Half-Life series®, Counter-Strike®, the Halo® series, etc.

All FPSs feature the core gameplay elements of movement and shooting, but many variations exist, with different titles emphasising certain aspects of the gameplay. The lines between sub-genres are not distinct; but all FPS on the PC utilise a combination of QWERTY keyboard and mouse as a means of controlling the game. Usually FPS control schemes are fully customisable within the game. One hand uses the mouse, which is used for free look (also known as mouse look), aiming and turning the player's view horizontally and vertically.

2.1 Historical evolution of FPS control scheme

The movement functions were the first functions that were mapped on the keyboard when the FPS games began to spread in the PC's in the 90's. Those functions were placed on the arrows keys (on the right side of the keyboard) and provided digital movement forwards, backwards, and turning left and right. This was also the way of aiming with the gun. There was no up-down feature so all the enemies were at the same level (eye-level). The demand for movement was still pretty basic and the "fire"

function was at the left Ctrl key so as to be used with the left hand. This was essentially a two-dimensional world where the player's avatar could move and shoot in a one level plane.

The situation changed completely as soon as the mouse appeared near the keyboard. New potentials were given to the game designers and the "free look" arose and stayed ever since. Free look (also known as mouse look) is a term that describes the ability to move the mouse to control the avatar's view in any type of computer and video games. Controlling this way the avatar's view, meant that the head became autonomous as far as the up and down view is concerned but the left and right view controlled also the orientation of the whole body. It was certainly one of the most significant technical breakthroughs of mid-1990s first-person perspective games.

However, the arrow keys seemed to face a number of design problems with this change. First of all, for the left hand it was hard to use them because of their position on the right side of the keyboard and secondly new functions later added such as jumping and crouching were difficult to be placed near those remote keys. So another breakthrough happened almost right after the introduction of the mouse look. The WASD mapping. Made popular by Quake®, WASD (also known as Was-duh, WSAD or ASDW) is a set of four keys on the left-hand side of a QWERTY or QWERTZ computer keyboard. W/S control forward and backward and A/D control sidewalk. This set of keys mimicked the arrow keys with the difference that they did not provide a turning feature, as the mouse now provided this.

Thus, The addition of the mouse look also meant that i) the movement functions jumped to the left hand (to the WASD combination of keys, ii) the keys for left and right turn which were not needed any more changed function providing an additional feature: sidewalk. This allowed for a separation between movement and the avatar's midsagittal plane's orientation taking a further step towards a more realistic game play. The above changes did not only significantly improve movement and shooting performance and accuracy, but had a more pervading effect in the design of next generation games by introducing a true three dimensional environment.

Also, the WASD placement contributed to a further development of FPS games. Additional functions were assigned to keys around the WASD for easy access by the left hand (e.g. Jump, Duck, Walk/Run etc). This positioning of the basic functions (more or less introduced with the release of Quake in 1995) remained stable until today with small variations among players.

2.2 The present design

Even though there is always some variation between different games and different players, modern FPS games share more or less the same general control scheme (Figure 1). This scheme might seem ad-hoc at first sight; however, if studied in detail it actually shows a deliberate effort to reconcile performance and ease of use subject

to the keyboard-mouse constraints. Although not optimum in any analytical sense, the adaptations, additions and alterations made over the years by developers and users have resulted in a non trivial result that seems to blend in an original way diverse and often conflicting ergonomic requirements. A number of examples are presented below.

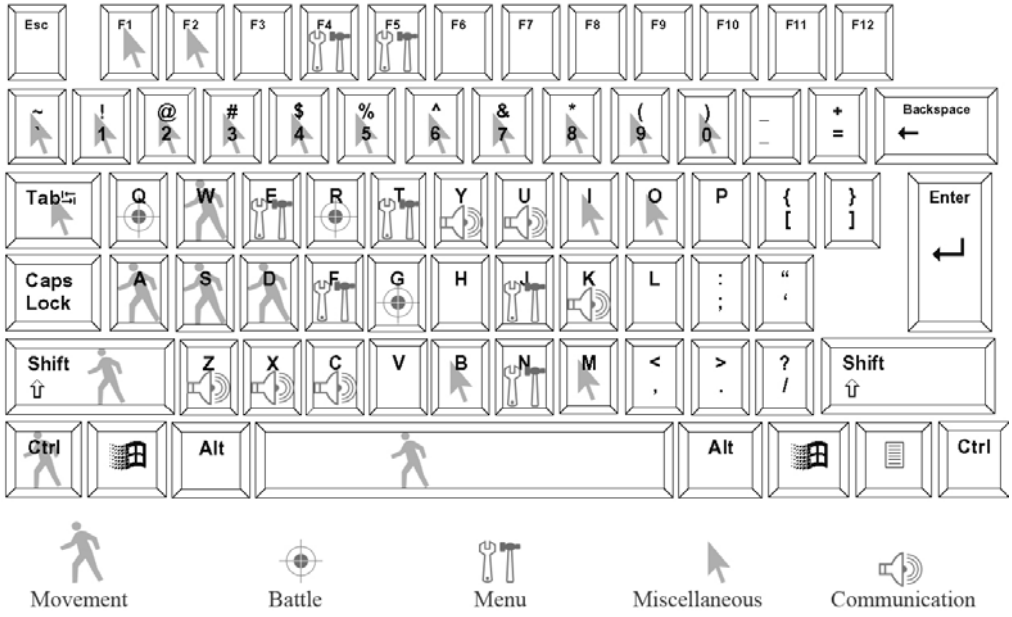

Figure 1. *The default keyboard mapping for the game Counter Strike Source®*

Consistency between controls and functions. The movement functions (forward, back, left, right) are mapped on the "WASD" – or in some cases "ESDF" – following both i) a spatial metaphor of the actual movements in the virtual ***world***, and ii) the topology of the three middle fingers on the keyboard (the middle being longer is assigned to W). These functions are used on a continuous basis (often in parallel) so the main interest of the designers was to enhance intuitive use and minimize finger strain. The switch from the arrow keys to the WASD did abandon the mapping at the semantic level but provided for the ability to use the large keys around the WASD area with the thump and little finger. This specific arrangement clearly favours sensorimotor appropriation and speed over compatibility at a reflective level.

Frequency - Speed of Use. At FPS games there is exigent need for instant responses and repetition at the use of certain features like the pull of the trigger and the secondary function of the gun (usually the collimator or an alternative fire option) As we saw above the fire option was first mapped on the left Ctrl as it was manipulated by the left index and its corner position was easily accessed haptically. At that time

there was no secondary fire option. The mouse of course changed the whole situation as the right hand took charge of the aiming and the two mouse buttons started being the primary and secondary fire functions. These changes are considered successful as i) the aiming through the mouse makes use of the most appropriated feature in GUIs i.e. the eye-hand coordination ii) the index and the middle finger are most capable of instant responses and repetitive strain. Also important is the task of switching between the carried weapons. At the beginning this function was performed by the numeric keys (1-9). This kind of mapping will be analyzed further down as it is considered primarily a semantic mapping. At the moment when a scroll wheel was added to the mouse though, the function of switching weapons was transferred there, but without being deleted from its original place. This redundancy in the switch weapon function enhanced learnability without cancelling speed of use.

Proximity and Physical shape-size. As mentioned above, an important reason for shifting the movement controls over to the "WASD" was the need for more functions around the movement keys. Therefore, some important functions such as JUMP, DUCK and USE were assigned to keys around the WASD. Such keys are Shift, Ctrl, Space and E that due to their size and position are easily accessed by the small finger and the thumb. Nowadays, these functions are rarely subject to variations from game to game probably due to their appropriation by experienced players.

Semantic mapping. The first use of this criterion has already been mentioned with the example of the numeric keys for selecting weapons. In that case the numbers on the keys correspond to the importance of the different weapons. Also, when additional functions were added (such as RELOAD, BUY and FLASHLIGHT) those not so often used functions were placed on proximal to WASD keys but whose letter symbols corresponded to the first letter of each function. So the keys R, B and F were assigned to Reload, Buy and Flashlight respectively acting mostly as a reminder for the player capitalizing on their respective symbols. Thus less frequently used functions tend to favour semantic mapping over other criteria.

Critical Error avoidance. Error avoidance is an important reason for mapping certain functions in places where the left hand can hardly reach. Thus, functions such as "Report Bug" and "Select Team", that disrupt the normal game flow, have been assigned to keys far from the working area (to F4 and M keys respectively) so as to prevent them from being accidentally pressed.

No criterion. Several functions that arouse lately (such as communication options) have been randomly placed on the keyboard. It seems that designers and users have not yet devised an inventive key arrangement for them in the remaining – not already assigned – keys of the keyboard. Functions such as "use voice communication", "spray logo" and "buy equipment menu" were assigned to keys under no specific criterion, and this is probably why they tend to be subject to miss hits and other inconveniences (e.g. effort to locate and discriminate between them).

The requirements given in the examples above are neither exhaustive nor mutually excluding. They form part of a preliminary analysis which needs further development based on empirical evidence of actual use.

A part of the list of assigned keys, functions, functional groups and criteria of assignment as judged by the authors is presented in Annex 1.

2.3 The co-evolution between games, gamer strategies and control schemes

From the analysis above it is evident that FPS game control schemes evolved with no overall plan. Indeed, at first sight the present input system looks like a compromise between, on the one hand the simplicity of using all-purpose input devices and on the other, an acceptable level of usability. This is true in a very pragmatic sense. At least for the movement control the left hand takes high strain, the three middle fingers (index, middle. ring) constantly upon the WASD, while the little finger and the thumb move up and down in order to control Jump, Duck and other functions. To a lay observer the above movement control scheme looks terribly awkward. However, almost all expert players prefer to use this instead of a dedicated controller (e.g. a gamepad[1]). An analytical HCI approach would suggest that it is easy to improve on it, and indeed it would be so if one decided to ignore that:

a) There is a large existing expert player community that has developed sensorimotor skills comparable to these of a musical instrument player or an expert typewriter. Actually, one important aspect of game satisfaction for these people is the challenge of achieving mastery in these skills [Clanton (1998)].

b) The diverse challenges that the designers implement in the games have been progressively tailored to match the characteristics of the current control scheme. In other words the control scheme as artefact has shaped the task (Norman 1991). For example, the ever-increasing number of functions in the games is most probably connected to the fact that the keyboard provides abundant keys for assignment.

c) The player community has historically shaped the games and continues to do so by experimenting with new tricks e.g. altering the control scheme, programming macro-functions etc. Such user interventions are in fact a part of the games' ability to provide satisfaction. This is evidenced by

[1] A recent example is the release early in 2007 of a device that lets users connect a keyboard and mouse on the popular Xbox360™ console. This caused great upheaval in the online Xbox™ FPS community as there is great concern that it gives players using it an "unfair advantage". Aiming with the mouse is considerably faster than with the analogue stick of the game pad and now the community is trying to find ways of dealing with this problem.

international FPS tournaments where such interventions are accepted as integral part of a player's skill [Tulathimutte (2004)].

The current FPS control scheme was never designed in the strict sense. It has emerged as a result of a mute dialogue between the players developing skills and interventions and designer induced challenges, subject to the keyboard – mouse constraints and affordances. For example when the mouse was introduced as the aiming and firing device, the pace and accuracy of the games had to de adjusted accordingly, in order to keep the challenge at an equal level as before. Also the adaptations progressively made by expert users to the interface (e.g. assigning different keys to functions as a result of game strategy) were retrospectively acknowledged by designers who re-calibrated challenge levels and challenge types accordingly [Tulathimutte (2004)]. We may then see the history of FPS games as a co-evolution between user expertise and intervention, control scheme configuration and the challenges inbuilt in the games' software.

The FPS evolution is in fact a paradigmatic case of the task-artefact cycle as introduced by Carroll et al. (1991). The task-artefact cycle captures the idea that tasks and artefacts co-evolve. The cyclical relationship between the two can be described as follows: A given task sets requirements for the design of an artefact to help in the performance of the task. The resulting artefact in turn, creates new or unexpected possibilities or poses new constraints on the performance of the task. These possibilities and/or constraints often suggest a revision of the original task for which the artefact was made [Norman (1991)], [Bannon & Bødker (1991)]. The new task sets new requirements for the redesign of the artefact and so on perpetually. The task-artefact cycle is in other words an iterative process of continuous, mutually dependent evolution between task and artefact [de Lëon (1999)], [Papantoniou et al. (2003)]. In the FPS case the task artefact co-evolution between game tasks and controller system is particularly vivid. This is due, on the one hand to the open-endedness of the games, which makes for frequent goal revision and search of alternative strategies, and on the other to the malleability of the keyboard mouse configuration.

A less malleable controller system, e.g. a game pad optimized for a specific game, would definitely be easier to learn and probably easier to use. However it would allow for player intervention only at the task side, greatly reducing one of the joys of the game (e.g. controller customization) not to mention eliminating a major driver for game innovation.

Indeed, a very important aspect in which the game pad differs from the mouse and keyboard is the lack of customizability. Generally the game pads leave no space for changes in key mapping, so players are obliged to follow the default mapping that comes with each game. On the contrary, in the PC customization options were always unlimited mainly because of the need to accommodate for the diversity of hardware used by each player. Customizability then, gave the opportunity for some

revolutionary ideas to emerge from the players' community, which in time formed the current control scheme design as a universal standard among FPS games. At present the control scheme although dominated by a quasi-stable key arrangement, continues to provide for customization by players (in many cases even promoting player customizations by facilitating the construction of macro-functions), so the evolutionary process is still active. For example today, with the multi-button mice being widely available commercially, expert players assign more and more functions to them in order to relieve the high load of the left hand. Wide adoption of the multi-button mouse will inevitably alter expert game-play both in terms of pace and tactics. This may trigger another cycle of co-evolution with developers adding new challenges and probably providing new functions (e.g. making use of the redundant function keys now assigned to the multi-button mouse).

The above situation allows advanced players to improve their performance and to reach a higher level of gaming. Thus, player contribution to the enhancement of the controllers was and still is significant. Game pads on the other hand allow for a fast learning curve, as they obey the ergonomic criteria inherent in their design. However, their rigidity prohibits creative interventions from experienced users, ultimately restricting performance, to a lower level of gaming.

It is not an exaggeration to suggest that FPS evolution would be considerably inhibited if the games were restricted by a dedicated purpose built controller. Be that as it may, one thing seems certain; the modern FPS would not exist as we know it today if it was not for the plasticity of the keyboard and mouse controller system.

2.4 Discussion

There is a pragmatic lesson to be learned from the evolution of the FPS with wider implications for the HCI community. The time when software applications could be considered as novel for all, has passed. Today in many domains we find highly experienced user communities with needs that evolve rapidly, and in a dialectic manner with successive versions of their software tools. In such environments, successful designs may not be the ones “optimized” for the here and now of specific applications or tasks. Designers need to acknowledge this in order to manage the co-evolution between i) users skills and strategies, ii) software and iii) interface features, judiciously allowing for plasticity. This way they prevent stagnation of their designed artefacts stimulating expert user inventiveness. In other words designers, by allowing for plasticity, allocate part of their activity to the user community, incorporating in a way part of the future design activity into the affordances offered in the designed artefact.

On the users’ side, plasticity may sacrifice learnability, and even performance up to a certain point, if taken in its narrow sense as time to completion of a pre-specified set of tasks. However, for an experienced and motivated user community these are minor

inconveniences compared to the opportunity to devise new ways of using the application and finally of having a say in its evolution.

The present paper is part of an engineering thesis at the department of Product and Systems engineering of the University of the Aegean aiming at the design of a dedicated controller for FPS games.

3. References

Bannon L. J. & Bødker S. (1991). *Beyond the Interface: Encountering Artifacts in Use.* In J.M. Carroll (Ed.), Designing Interaction: Psychology at the human-computer interface. New York: Cambridge University Press, ISBN 0521409217

Carroll, J. M., Kellogg, W. A., & Rosson, M. B. (1991). *The task-artefact cycle*. In J. M. Carroll (Ed.), Designing interaction: Psychology at the human-computer interface. New York: Cambridge University Press, ISBN 0521409217.

Clanton, C. (1998, April). *An Interpreted Demonstration of Computer Game Design.* Proceedings of the conference on CHI 98 summary: human factors in computing systems. Chi 98, 1–2.

De Léon, David (1999): *Building Thought into Things*. In: Proceedings of the 3rd European Conference on Cognitive Science. 0000. p.37-47.

Gee J. P. (2003). What Video Games Have to Teach Us About Learning and Literacy. Palgrave Macmillan, ISBN: 1403961697

Grice, Roger. (2000). *I'd Rather Play Computer Games Than Do Real Work! (Wouldn't you?): The Appeal and Usability of Games Interfaces*. Make It Easy 2000 Conference, IBM.

ISO 9241 (1998). *Ergonomic requirements for office work with visual display erminals (VDTs) – Part 11: Guidance on usability*

Koster, R. (2004) *A Theory of Fun for Game Design*. Paraglyph Press, ISBN 1932111972

Norman, D. A. (1991). *Cognitive artifacts*. In J. M. Carroll (Ed.), Designing interaction: Psychology at the human-computer interface (pp. 17-38). Cambridge: Cambridge University Press, ISBN 0521409217.

Papantoniou, B., Nathanael, D. & Marmaras, N. (2003). *Moving Target: Designing for Evolving Practice*. In Universal Access in HCI: Inclusive Design in the Information Society, C. Stefanidis (Eds.), Mahwah: Lawrence Erlbaum Assoc., Vol. 4.

ANNEX 1. *Example of Assigned keys, functions, functional groups and criteria of assignment*

Key	**Function**	**Group**	**Criteria**
MOUSE MOVEMENT	Mouse look	MOVEMENT	*Frequency - Speed of Use*
W	Move forward	MOVEMENT	*Consistency*
S	Move back	MOVEMENT	*Consistency*
A	Move left (Strafe)	MOVEMENT	*Consistency*
D	Move right (Strafe)	MOVEMENT	*Consistency*
SHIFT	Walk (Move slowly)	MOVEMENT	*Proximity and Physical shape-size*
SPACE	Jump	MOVEMENT	*Proximity and Physical shape-size*
CTRL	Duck	MOVEMENT	*Proximity and Physical shape-size*
MOUSE1	Fire	COMBAT	*Frequency - Speed of Use*
MOUSE2	Weapon special function	COMBAT	*Frequency - Speed of Use*
R	Reload	COMBAT	*Semantic mapping / Frequency of Use*
MWHEEL UP	Previous weapon	COMBAT	*Frequency - Speed of Use*
MWEEL DOWN	Next Weapon	COMBAT	*Frequency - Speed of Use*
Q	Last weapon used	COMBAT	*No criterion*
G	Drop current weapon	COMBAT	*Critical Error avoidance*