B.L.A. – Body Language Application

David Simão
Instituto Superior Técnico / YDreams
david.simao@ist.utl.pt

Jorge Pereira
Instituto Superior Técnico / YDreams
jfsp@ist.utl.pt

Carlos Martinho
Instituto Superior Técnico / GAIPS
carlos.martinho@ist.utl.pt

André Almeida
YDreams
andre.almeida@ydreams.com

Resumo

Apesar das diferentes formas de interacção com computadores possíveis hoje em dia, os videojogos encontram-se ainda bastante vinculados aos controladores tradicionais. Neste paper exploramos a detecção de movimentos como forma de interacção num jogo cooperativo. Utilizando a plataforma YVision da YDreams[1], aplicámos tecnologias de visão computacional para desenvolver um jogo que desafia dois jogadores imersos num ambiente virtual a coordenarem os seus movimentos para atingirem um objectivo comum. A avaliação realizada sugere que o modo como os utilizadores interagem com a aplicação, combinado com um clima de cooperatividade, afectam significativamente a sua experiencia de gameplay.

Abstract

Although different possible modes of human-computer interaction exist, today's video games are still pretty much connected with traditional controllers. In this paper, we explore movement detection as a way of interaction in a cooperative game. Using YDreams' YVision framework, we applied computer vision technologies to develop a two player game, in which they coordinate their movements to achieve a common objective. Evaluation suggests that interaction and cooperativity affect significantly the gameplay experience.

Palavras- chave : *Interacção, Detecção de Movimentos,Visão Computacional, Virtualidade Aumentada*

Keywords: *Interaction, Motion Detection, Computer Vision, Augmented Virtuality*

"The most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it."

Mark Weiser

1. Introduction

The history of game industry can be divided into a series of generations. Each generation begins when new hardware, typically more powerful than its predecessors, is released (Marshall, 2006). If you ask any gamer what they like best about video games nowadays, there is a good chance the answer will be related either to graphics, or gameplay. While those two criteria are very important to the creation of good video games, we often ignore and neglect the main aspect that determines the way we play: game controllers (Koushik, 2006). Since the dawn of video games, we've seen controllers ranging from button pads to the most advanced technologies of human-computer interaction (HCI).

Current game consoles, like Microsoft Xbox 360[2] and Sony Playstation 3[3], continued the "more is better" approach to controller design. Game controllers now have multiple buttons, confusing layouts and different directional navigation directions. These schemes place barriers between the player and the game. For a casual gamer, modern controllers can be intimidating. Not only must they learn to play the game, but also achieve a certain level of dexterity to use these controllers (Marshall, 2006).

Today's video games are moving towards several new interaction paradigms. Due to the growing computational capacity of computers, it is possible to interact with applications in a more natural and intuitive manner, increasing the player's sense of presence in the virtual world. The swing of an arm replaces the push of a joystick, spoken commands replaces synchronous button presses. Young and old players can compete on a level playing field regardless of gaming experience (Marshall, 2006).

In this paper we present our proposal of a natural interaction video game for two or more players, which targets to create a short term, but intense and fun gameplay experience. We start by referencing several state-of-the-art technologies used in controller less video games, which served as base and inspiration. Then, our game concept is presented, followed by the development process, including a description of the YVision framework, the platform used for implementation. In the last sections, we describe the playtesting experience and discuss its results that led to most of the conclusions and plans for future work.

2. Projects and technologies that inspired us

Despite being left aside of video games for a very long time, the role played by HCI is raising its importance in this area. Bad player-game interfaces often decrease amusement and the sense of immersion in the game world. Sometimes they can even result in misunderstanding of the designer's intent. The continuously growing evolution of HCI, namely in vision based interfaces is proving to be a major step in breaking the so-called barriers between the players and video games. Vision can be a powerful tool in interface design for video games because of its potential for sensing the players' physical utterances like position, orientation, gestures or movements.

Among the various possibilities for interface designs, we studied four different groups, that served as base and inspiration for our work: Gesture Recognition is a technique that consists in detecting specific player movements and convert them to actions in the game world; Motion Detection, the technology behind the interface presented in this work, which consists basically of detecting movement and directional flows in a sequence of images; Hotspot Triggering, used when we want to activate specific 2D areas of an image with parts of our body; Full Body Interaction, as the name suggests, consists of controlling a computer application, by making use of the entire body.

Gesture Recognition

Human gestures promise to be a very powerful tool in HCI. Typically, gesture based interfaces establish a correspondence between a human gesture and a command in the computer application, as a replacement for keyboards, mouse or gamepads. There are two different types of gestures that can be used for recognition: 2D and 3D.

2D gesture recognition is usually associated to traditional computer vision technologies, which is still facing several challenges in the field of interactive multimedia (Freeman 98). Freeman defends that video games require a very fast response time which is not only highly dependable on the recognition algorithms but also on the hardware's speed (capture rate and image processing), in order to avoid delay between real and virtual actions. Though, state of the art cameras and processors already provide very acceptable capture and processing frame rates, the required conditions to create interfaces with short response times. Another challenge pointed out by Freeman, is the disambiguation of gestures: "Computer vision algorithms should be reliable, work for different people, and work against unpredictable backgrounds". Kang et al (Kang 2004) address this challenge with a system that combines gesture spotting and gesture recognition to separate unintentional movements from meaningful movements. While the system proved high reliability when applied to the game Quake II[4], we believe that vision-based gesture recognition systems still don't apply for games that require high levels of accuracy.

3D gestures have integrated HCI research since "Put that there" (Bolt 80). These interfaces have been using cameras since the first approaches. However, accelerometers are raising popularity in spatial gesture recognition. Recent projects like the Wiizards (Kratz 07) make use of Wii-mote, the the wireless controller that comes with Nintendo Wii[7] to generate motion paths and classify them as actions, using Bayesian networks[5]. Accelerometers can provide greater fidelity for gesture tracking, however when compared to cameras, these devices are considered more intrusive, because users actually have to carry them. Another challenge in 3D gesture-based interfaces is the fact that accelerometers alone cannot provide the user's spatial position, which is the reason why most accelerometer-based interfaces include infra-red or z-axis cameras.

Hotspot Triggering

Hotspots are defined as rectangular areas in the captured video image that can be activated, according to predefined activation constraint. Typically a hotspot is activated, when the system detects motion in the respective 2D region. Activation of a hotspot is used as a trigger for a specific command in the application. Simple games like Eye Toy Kinetic[6] represented in Figure 1, can be created using hotspot triggering. In this game, one of the objectives is to hit

several targets on the screen. Each target is assigned to a hotspot, whose activation results in destruction of the target.

Figure 1 - Eye Toy Kinetic. The user destroys the targets by touching hotspots in the image. Source: Downloaded from http://www.dignews.com/platforms/ps2/ps2-news/sony-gets-kinetic-headline/.

The main problem of using this technique, as well as other vision-based techniques is that they are very prone to errors, as the camera image has a large amount of information, which needs to be filtered in order to avoid errors and unintentional actions. Jaimes and Liu (Jaimes 05) address this problem by creating skin filters and activation constraints in the hotspots, making them more robust, as they can separate the user's hand from the background and control which motion vectors are eligible to activate them.

Using different combinations of hotspots and constraints, it is possible to achieve great variability in the application. However, hotspot triggering can provide higher levels of control when combined with other interaction techniques.

Motion Detection

Motion detection is probably the most used technology in video games that involve human motion. Often, a player's motion signals are important information to the application (i.e. bending the controller when making a turn in racing games). There are several ways of detecting motion in computer applications. Video cameras are probably the most used, although accelerometers and motion sensors also serve the purpose. The Wii-mote, is currently one of the most popular devices, in game industry, that can be used for motion detection, alongside with Apple iPhone[8] and iPod Touch[9]. These devices include built-in

accelerometers that provide orientation and velocity information, which can be used to detect and identify gestures or simply motion. However, commercial games started to use motion detection long before the Wii-mote. One such example is the EyeToy Camera[10] , mentioned above, which allows the player to interact with games using motion, color detection and also sound.

Once the context of the application is known, it is easier to convert motion information into game actions. Freeman (Freeman 98) uses motion analysis to control the Sega Saturn game Decathlete. While the handheld control can be a major restriction as an interface of an Olympic sports video game, human motion proved to be very efficient. The authors use optical flow analysis to convert the players' movement into game actions.

Computer games aren't the only research matter where movement detection plays a significant role. TinyMotion (Wang 2006) is a middleware application for mobile phones that detects the user's hand movement through a built-in camera, allowing the user to control his phone using gestures. As video cameras and accelerometers begin to slowly integrate almost every mobile device, the amount of commercial applications that make use of such assets for motion detection is also rising.

Full Body Interaction

One of the first attempts to decouple players from controllers, and use their whole body to interact with video games was the ALIVE system (Maes, 1997), which allows full-body interaction with virtual character agents. The user's image is captured and placed in a virtual world, and that works as an augmented mirror, representing the player surrounded by virtual objects and agents.

More recently, Jonah Warren (Warren 2003) developed an interactive vocabulary, to aid in the creation of full body based video games, and also an interactive installation, composed by a computer, a camera and a projector, allowing users to choose between three different games that make use of different body actions like crouching and jumping.

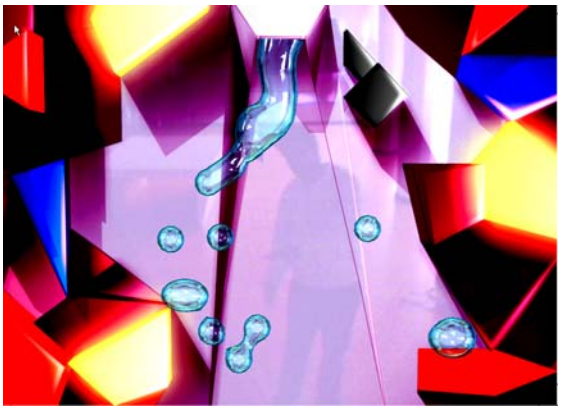


Figure 2: The Waterfall application. Users stand in front of a camera and the image is placed as an overlay under a virtual waterfall. By moving around, the user is able to interact with the water.

YDreams uses the same hardware setup to build full body interactive applications like the Waterfall (Figure 2). In this application, like in Warren's work, users stand in front of a camera which mirrors their image in a 2D projection that allows them to interfere with a waterfall by placing obstacles and interrupting the water flux with their own body.

Full body interaction is slowly taking its first step into game industry: Microsoft already presented their research project in the field: “Natal” for the Xbox 360 which enables user to control and interact with the gaming platform through full-body 3D motion capture, voice commands or presented objects and images (Graft, 2009).

3. The Game

B.L.A. stands for Body Language Application, a game which allows full body interaction. This approach promotes the use of physical exercise as a replacement for controllers in video games. The interaction is done in real-time with movement replacing the use of game controllers and making the game experience enjoyable to a very diverse set of players that usually don't play video games.

The game's logic is centered on a non-animated vehicle: a pump-trolley that runs on rail tracks, which is fueled by forces generated by at least two players (one in each side of the crossbar, Figures 7 and 6). This association between reality and virtuality turns out to be described by motions: each user jumps to generate power on his side of the cart. Based on this momentum, the cooperation between two players to jump in an alternate way becomes

essential to move forward and progress in the game. The cooperativity requirement (one player alone isn't able to move the trolley), makes this a social game, and increases player engagement. The higher the cooperation, the greater the speed players achieve, which is rewarded with a special bonus. In addition to alternate jumps, there are synchronized jumps that make the trolley jump and are needed to avoid obstacles.

Despite focusing on the interaction with users, we didn't want to leave other areas unexplored. The scenario is also important because it's our way of giving feedback to the user. Thus, we created a colorful world which conveys a sense of fun and less serious gaming (Figure 4), giving users the freedom to fail and laugh. However, to give a better feedback, a sense of immersion was created through the augmented virtual space: the users' image cut and placed in the right positions of the trolley to obtain an extra motivation and to increase the sense of presence.

Fig 3: Users playing BLA at MoJo

Fig 4: A snapshot of the game in action

4. Development

In this section, we describe the key concepts of the development framework used in this project. YVision (Almada, 2009) is a general purpose programming framework developed by YDreams to create data-driven applications, which enable fast prototyping and development. Being designed on a multimedia context, YVision integrates several different technologies each one addressing a distinct challenge, such as rendering, physics simulation or image processing. These technologies are built in independent modules, and implemented above the YVision core, making it a completely generic architecture.

Graphs and Workflow

Workflows are a common way of representing computer applications with a sequence of delegated tasks. The YVision Graph framework explicitly enables the use of a workflow abstraction to build applications. This programming paradigm focuses on isolated functionalities. Each workflow task constitutes a block, which is decoupled from the rest of the application, so it can be reused for multiple purposes. Blocks are simply black boxes, with input, output and properties (custom parameters, specific to the block's activity). Complex application scenarios can be achieved by connecting several blocks using data flow connections. From a high-level perspective, BLA is a workflow, composed by a sequence of connected blocks, which execute image processing and world simulation tasks.

World Objects

One of the main concerns when designing a video-game is the definition of the game object, i.e the objects that constitute the game world. In the YVision framework, those objects are modeled as component-based architectures (Stoy 06, Rene 06, Garces06). World objects are service (component) containers, whose single purpose is to access and manage the object's own services. The overall object behavior and characteristics are defined by the properties and interactions between those services. This allows us, for example, to define a world object for representing the pump-trolley specifying not only all its physics characteristics but also his controller actions, using customized services.

Services

Each service represents a specific characteristic of an object. Supposing there's an object with a visual representation in a 3D world: our object will necessarily contain a service responsible for keeping the data and behavior (code) needed to enable that representation. Services are developed independently from the object they are placed in. Communication between services is implemented using special components, which have access to the object's service collection and the services' interface called Behaviors.

Behaviors

Behaviors layout the rules for the object's action in the world. They implement the object's functionality, while services store the object's information. Behaviors are defined as a common invocation and execution interface and usually take up the role of combining the information in distinct components of a world object. Each behavior is a latent task, since it can run during several application cycles and can either succed or fail. Complexity in world objects is then achieved by hierarchically composing behaviors in tree structures (Champandard 08). Every action that happens in our application, is executed by a behavior. The pump-trolley, for instance owns customized behaviors, which enable it to move forward and backwards, and correct the speed according to the players' performance.

Settings

In order to run with accurate performance, several multimedia applications require calibration parameters like the FOV in videocameras for instance. When developing an application in YVision, users can easily associate each calibration parameter to the object it belongs by annotating it as a metadata field. Developers are then able to easily create graphical user interfaces (GUI's) for adjusting calibration parameters in runtime, by using YVision's general configuration framework: YVision Settings. This proved to be one of the most useful characteristics in our application, since many settings like camera calibration need to be adjusted to the environment. By using the settings framework, we were able to setup the application parameters in runtime, even after the players started to interact with the application.

Base Application: Interactive Surfaces

Interactive surfaces (IS) is an application developed using the YVision framework, and on top of which we built our game. The advantage of using IS as the bootstrap for Body Language Application is that we were saved the time of projecting and implementing an application core. IS has its own workflow, which provides users the basic steps of developing interactive multimedia applications: capture user inputs, detect features on those inputs, interpret those features, and simulate a virtual world. From a high-level perspective, we can say that our application is no more than a specific customization of Interactive Surfaces.

Game Architecture

In order to get the users to control a virtual avatar in the real world, we use a pipeline with a video capture module, several information processing tasks and a world simulation module. The camera on Figure 6 is encapsulated in a capture module that places the camera frame sequence in the workflow. Those images are then processed by two subsequent modules (Background Subtraction and Motion detection). Once all features are detected, the system places them in world objects, and interprets them as game actions which are then processed by the simulation of the virtual world like most common games.

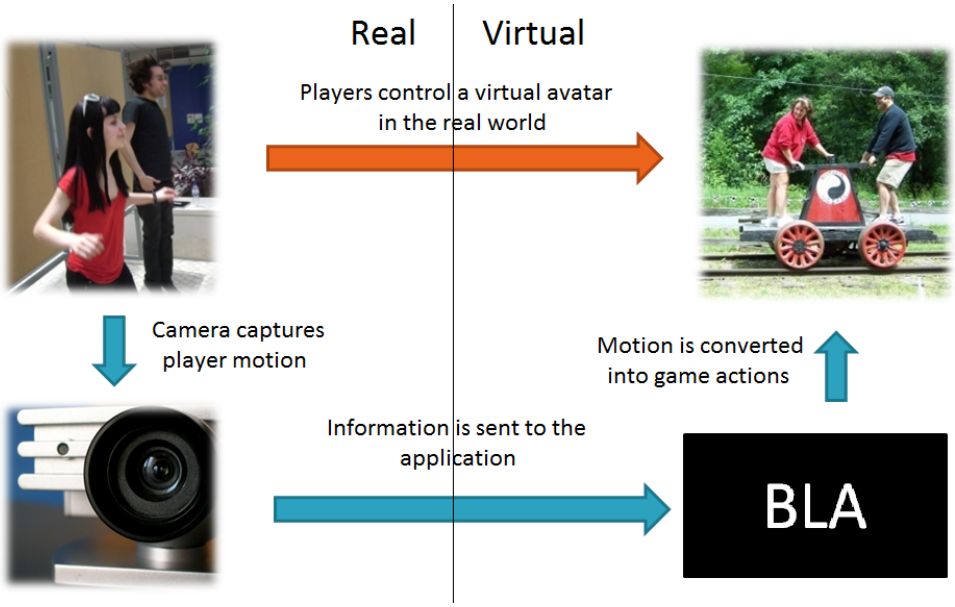


Figure 6: BLA interaction and simulation pipeline. The orange arrow represents the interaction process as perceived by the players. The blue arrows represent the real information flow. Source: Camera and Trolley images downloaded from http://www.flickr.com

Pump-Trolley

In order to achieve a realistic interaction, we decided to simulate the physics of a real pump-trolley. We built a physics model, based on simple primitives (boxes and cylinders) that actually moved forwards and backwards by applying pressure on the crossbar. To assemble the structure, we defined key points which ensured the overall base functions of the trolley (gray circles on Figure 7). In the circles with horizontal lines (Figure 7) we used hinge joint[11] type constraints. Once we got the trolley running, we needed to be sure that the car wouldn't change directions (just like a train on tracks). Because we couldn't afford the cost of

simulating real train tracks, we locked rotations in the y axes on the locations marked by the circles with a mesh inside (Figure 7).

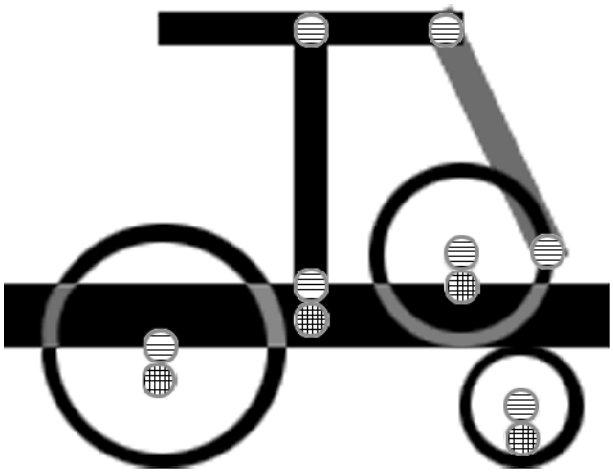


Figure 7: Vehicle chassis assembled with Hinge Joints (Horizontal lines) and Angular Motors (Mesh)

Player Avatar

Augmenting the virtual environment was also a major challenge. Since players are being filmed by a video camera, we have all the information we need to create a mirror projection of the user where he can see his real image holding on the virtual crossbar of the pump-trolley. So what we did was to use a background subtraction (Bradski, 2008) module to outline the player's silhouette. Then, we masked the actual camera frame with the outputs of the background subtraction (Figure 8), obtaining a silhouette of the player that we can use as a texture in the virtual world.

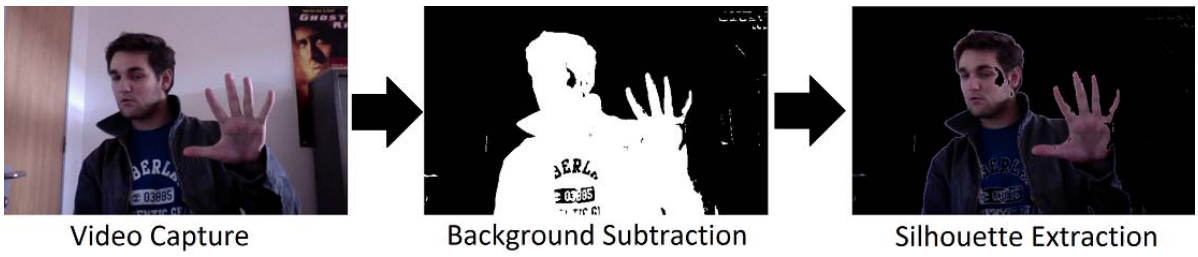


Figure 8: Silhouette extraction pipeline

Interaction

To convert the users' movements in the real world to game actions we used a motion detection module. This module detects the optical flow of the image sequence captured by the camera. In other words, it creates vectors with the same direction of the movement in the image sequence (Figure 9). We are particularly interested in the vertical ones, so we project every vector in the y axis. Then, because we use only one camera, the image is divided in two and the motion vectors on each half are summed up for each player. Finally, we take each sums vector and use it as a force to apply on the crossbar's left and right ends.

Figure 9: Motion vectors detected, based on optical flow extraction from an image sequence.

There is a direct correspondence between the size of the jump and the force applied to the trolley. Somehow we had to distinguish the efforts of users on different jumps, so a coefficient is applied to the size of the users' movement and then the force is applied according to that value, which may benefit the coordination between two users or benefit from a higher trolley's jump enabling them to easily overcome an object. When no interactions are detected, we create a dynamic rudder force to stop the trolley from running uncontrolled, and to match the vehicle's movement with real player actions.

To address unintentional movements, we created a set of tests during development, which led us to optimal threshold values, which we use to filter motion vectors so the users are less likely to lose control of the trolley.

5. Playtesting

This section begins with a description of our approach to a playtesting experience for the BLA game. Next, we evaluate the most relevant points highlighted during the session and comment about possible improvements that can be introduced. In short, since the type of interaction of our application is quite different from common video games, the tests were based on the usability of the interface and the entertainment provided to players. For this purpose, we defined measures such as reaction time, frame rate, and player amusement, which we analyze further in this document.

Strategy

The first thing we wanted to evaluate was if players wanted to get to the end of the level, and, after that, playing again, or if they got tired and abandoned the game before the end of the level. We also wanted to understand if players started to create winning strategies, and last, we were concerned with usability questions such as if they understood how the game is played.

BLA was presented at a game workshop in Instituto Superior Técnico (Montra de Jogos). The thirty-eight testers were mainly students from different areas, some of which knew nothing about game development. The evaluation allowed us to obtain opinions from both academic and consumer perspectives.

The session started with a small description of the application context. Then, the players tried an introductory level with on-screen help (and sometimes our intervention when needed). The next levels were played without any intervention, and after playing, the testers answered a small survey. During the experience, we took notes and most of all, watched the players' reactions to the game (e.g. facial expression and verbal comments by the subjects).

Game usability

At the beginning, players were a little confused. We had to explain and intervene in almost every session. People who saw others playing before had less trouble getting started. In addition, when people finally got the grips on the game, obstacles appeared, and they had to interrupt their coordination. It's not common to see a pump-trolley jumping, so it's not

completely obvious that they need to make it jump (41% of the users only understood after some explanations). Our first conclusion was that an introductory level wasn't enough for people to understand the game mechanics. Most of testers don't know how a pump-trolley transmission works and consequently, can't figure out when is the exact moment to apply forces on their side.

Playing Strategies.

There were three major strategies used to achieve completion. The first one was the original intention of this concept: Coordinate movement to gain speed and jump when facing obstacles. The second strategy consisted of keeping up the highest speed possible, ignoring obstacles, so they could maintain the super speed bonus up. That was something that we were already expecting since the bonus points are worth it, and also because of the yellow sparks appearing from the exhaustion pipe. The last strategy was a complete surprise, and though it happened only once, we found it out quite interesting: two players jumped constantly at the same time. Not only did they manage to avoid most of the obstacles, but they were also able to maintain a constant speed due to the fact that there is no aerodynamic drag.

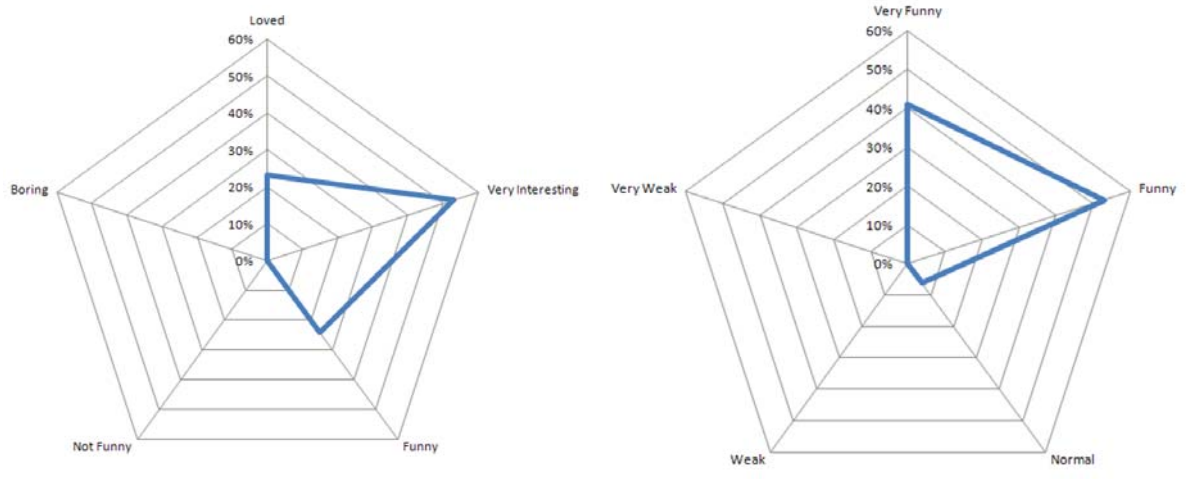
The lesson learnt for the next iteration of development is to create some physical drawbacks when colliding with an obstacle (e.g. slowing down), instead of reducing score only, and reduce the duration of jumps. However these small changes alone do not balance all the playing strategies. The difficulty degree should be similar, in other words, whatever strategy the player chooses, he should always find difficulties that challenge and motivate him to play again. Both the constant jumps and the speed bonus strategies become attractive to use, because of the few player actions they require and the simplicity of level s(only obstacles on the ground, no rail change or aerial objects). To fight the use of these strategies and get players to use all the available actions, one should create more diverse levels and obstalcles, for example forcing the player to change to another rail to avoid the end of the rail. This way, it is possible to increase the dependence of ground actions, and consequently, the jump dynamics would not be so obvious.

Interaction types.

The most common interactions were many small jumps in one pass, so lots of small forces were applied continuously to the crossbar. This revealed some disadvantages when users tried to switch roles. Some trolley jumps were misinterpreted with the final and the initial phase of the jump of each player (i.e. whilst being both still in the air). Yet, came what we called the "big jumpers", who instead of jumping multiple times, preferred to jump one big jump, thus applying a greater force on the crossbar. This appeared to be the easiest way of coordination. Other groups chose not to jump at all and just crouch. This interaction proved itself to also be effective, but not so entertaining. Last, to include some more discrete users in our tests, we encouraged them to try moving their hands up and down, creating motions which also successfully worked, but not so efficiently. This proved to be an alternative interaction form and opens the span of users to players with motor disabilities.

General observations.

Running the application on an Intel Core 2 Duo, with 2 Gb Ram and a GeForce 8400 GS graphics card, allowed us to achieve a capture resolution of 320x240 at the average frame rate of 30. When asked about the application reaction time, most people mentioned the dual jumps, which sometimes are not immediate due to a error tolerance. Although, most users were a little shy about jumping in public, we were happy to see some laughing and smiling going on during the game experience. In general, users had fun while playing BLA, even if the application didn't respond immediately (Figure 10), or if they didn't achieve success.



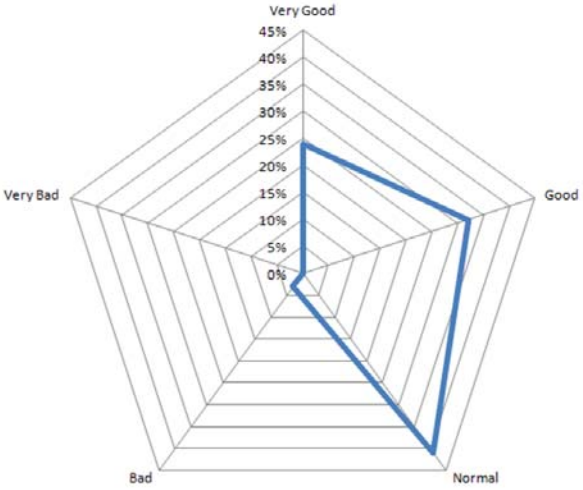


Figure 10: Users' global opinion (up left), Game amusement (up right), and Application reaction time.

Even when the vehicle crashed against obstacles or when users were totally uncoordinated, sending the trolley backwards, users want to try again, and their answers to the survey suggest that BLA is able to create a short, but engaging and entertaining experience.

6. Concluding Remarks

In the context of controller less interaction, our results suggest that is possible to create entertaining experiences by using motion detection. We also validated the fact, that a game like BLA, when played without the use of game controllers can be a lot more fun.

However, motion games do not sustain long-term game play because they lead to physical exhaustion of players. That is why games like BLA should be designed as short and casual experiences. Thus, we believe that more actions could be added, in order to increase the player's possibilities and create more intense experiences. Since we are using a video camera, a lot more information can be extracted besides motion vectors.

7. Future Work

According to what was said above, we will focus our future research, on how to make better use of the existing information to enrich the user experience and create additional actions such as changing directions or crouching to avoid aerial obstacles.

We also want to explore the possibility of creating different levels and objectives in such way that players can feel a rhythmic sequence, and play according to the rhythm. In (Csikszentmihalyi 1990), the author defends that rhythmic actions help the player to reach a state of "flow" and heightened concentration in games. Rhythmic obstacle sequences create a rythmic sequence of actions, and consequently player movements, not only helping them to easily reach a state of coordination, but also achieve greater success in bypassing obstacles. Previous work (Compton06, Smith 09) has proved that is possible to create fully playable levels, using rhythm-based content generation in 2-D platformers. We believe that such methods apply to the gameplay style in BLA, and can be a major contribution to increase player engagement during the experience.

References

Almada, Antão and Lopes, Gonçalo and Almeida, André and Frazão, João and Cardoso, Nuno, YVision: A General Purpose Software Composition Framework, (2009), Human Computer Interaction, New Trends, Springer Berlin / Heidelberg, Vol 5610/2009

Bolt, R. A. "Put-that-there": Voice and gesture at the graphics interface, (1980), Proceedings of the 7th Annual Conference on Computer Graphics and interactive Techniques (Seattle, Washington, United States, July 14 - 18, 1980). SIGGRAPH '80. ACM, New York, NY

Bradski, Gary and Kaehler, Adrian, Learning OpenCV: Computer Vision with the OpenCV Library (2008), O'Reilly, Cambridge, MA

Casamassina, Matt, IGN's Nintendo Wii FAQ , (2009), IGN, 24th June 2009, Online accessed, 8th March, 2010

Champandard, Alex J, Behavior trees for the next-gen game AI, (2008), Online accessed 10-

January-2010

Compton, K. and Mateas, M. Procedural Level Design for Platform Games (2006). Proc. 2nd Artificial Intelligence and Interactive Digital Entertainment Conference (AIIDE '06), Stanford, CA.

Freeman, W.T. and Anderson, D.B. and Beardsley, P.A. and Dodge, C.N and Roth, M.W.C.D. and Yerazunis, W.S., Computer vision for interactive computer graphics, (1998), IEEE Comput. Graphic Appl. 18, pp. 42–53

Garces, S. (2006), AI Game Programming Wisdom. Volume 3., CharlesRiver Media, pp. 251-265

Graft, Kris and Sheffield, Brandon, (2009), Microsoft's Future Begins Now, Gamasutra, 24th June 2009, Online accessed, 8th March, 2010

Jaimes, A. and Liu, J., Hotspot components for gesture-based interaction, (2005), IFIP Interact, Rome, Italy

Jaimes, A. and Liu, J., Configurable Hotspots for Ubiquitous Interaction, (2005) 7th International Conference on Ubiquitous Computing (ACM Ubicomp), Tokyo, Japan,

Kang, Hyun and Lee, Chang Woo and Jung, Keechul, (2004), Recognition-based gesture spotting in video games , Pattern Recognition Letters, Vol 25, Issue 4, page 1701-1714

Koushik, Sud, (2006), Evolution of Controllers, Kombo, 24th June 2009: http://wii.kombo.com/article.php?artid=6355, Online accessed, 8th March, 2010

Kratz, L., Smith, M., and Lee, F. J., Wiizards: 3D gesture recognition for game play input, (2007), Proceedings of the 2007 Conference on Future Play (Toronto, Canada, November 14 - 17, 2007). Future Play '07. ACM, New York, NY

Maes, Pattie and Darrell, Trevor and Blumberg, Bruce and Pentland, Alex, April, The ALIVE system: wireless, full-body interaction with autonomous agents, (1997), Computer Animation 95 Proceedings, Geneva, Switzerland

Marshall, Damien and Ward, Tomas and McLoone, Séamus, From chasing dots to reading minds, (2006), Crossroads (ACM), Vol 13, Issue 2, page 10-10.

Rene, B (2006), Game Programming Gems. Volume 5, CharlesRiver Media, pp. 25-37

Smith, G., Treanor, M., Whitehead, J., and Mateas, M., Rhythm-based level generation for 2D platformers. (2009), Proceedings of the 4th international Conference on Foundations of Digital Games (Orlando, Florida, April 26 - 30, 2009). FDG '09. ACM, New York, NY

Stoy, C. (2006), AI Game Programming Wisdom. Volume 6., CharlesRiver Media, pp. 393-403

Wang, J and Zhai, S. and Canny, S., (2006),Camera Phone Based Motion Sensing: Interaction Techniques, Applications and Performance Study. Proceedings on the 19th annual ACM Symposium on User interface software and technology, Montreux, Switzerland

Warren, Jonah. Unencumbered Full Body Interaction in Video Games, (2003). Master's Thesis, MFADT Program, Parsons School of Design,

Endnotes

1 YDreams Informática, S.A. http://www.ydreams.com

2 Xbox 360, Microsoft, 2005

3 Playstation 3 , Sony, 2006

4 Quake II, ID Software,1997

5 Bayesian Networks, http://en.wikipedia.org/wiki/Bayesian_network

6 EyeToy: Kinetic, Sony, 2005

7 Wii, Nintendo, 2006

8 iPhone, Apple, 2008

9 iPod Touch, Apple, 2007

10 EyeToy. Sony Computer Entertainment America, 2003

11 Hinge joint - http://en.wikipedia.org/wiki/Hinge_joint