Received December 20, 2016; Accepted December 30, 2016

International Journal of Asia Digital Art&Design

Sidestep and Sneak peek: spatial actions in augmented reality games

ARSurface - Dynamic Spatial Augmented Reality for Tangible Interface

Sudario, Anderson B.
Kyushu University
Fukuoka, Japan
absudario@gmail.com

Tomimatsu, Kiyoshi
Kyushu University
Fukuoka, Japan
tomimatu@design.kyushu-u.ac.jp

Abstract
We propose a novel concept of game, which allows players to experience tangible interaction with the virtual world of digital games by mixing motorised scenery with dynamic projection mapping.
A specific hardware - which includes a controllable platform, on where players can setup customised polygonal shaped scenery to play games projected onto its surfaces, and a single, or a pair, of focus free laser pico-projectors pointed toward the platform direction - is proposed as a game console able to run this experiment. The controllable platform orientation is synchronised with the game play in a way the moving physical scenery and the projected virtual contents are constantly aligned. By designing 3-dimensional (3D) animations, which are rendered and projected in accordance with the physical surface orientations, we were able to enhance the illusion of depth toward these planar projection during the game play, while giving the chance for our flattened 2-dimensional (2D) main character to make use of all directions which surrounds him.
The use of projection mapping is justified since we intent to exploit the characteristics of projected 2D light onto 3D objects in order to extend digital games means of expressions. In this paper we discuss about spatial actions, which are actions players can perform at physical object creases. When synchronised with motor movements, these actions extend the sense of volume in relation to the game character providing a tangible connection between players and digital content. We also discuss about the technical aspects regarding the development of this project and its application as a tangible game design tool.

Keywords: game design, tangible interface, projection, mixed reality

1 Introduction

The term Augmented Reality (AR) refers to the possibility of extending our perceived reality (real world) by adding computer-generated imagery to complete or replace physical objects we see. This mixed reality can be exploited in many ways, whether by making use of head-mounted or hand-held devices, resulting in individual AR experiences, or by overlaying, displaying or projecting virtual contents on the surface of physical objects, resulting in a collective AR experience [1]. Spatial Augmented Reality (SAR, also known as projection mapping) is a current trend on developing all sorts of mixed reality interfaces from artwork installations, experimental movies or games [2,3,4]. The use of projectors makes it possible to add new layer of virtual contents displayed on physical objects. Because of this characteristic, SAR can be easily associated with tangible user interface as a method to translate between shared virtuality and reality.

Interaction with spatially augmented objects is however, in many cases, limited to a setup where objects and projectors remain static in relation to each other or present some lack of mobility. Our research is motivated by exploiting techniques capable of producing adhesive SAR to be laid on top of non-static objects. Here we discuss about the piece of hardware developed in order to run our experiment with spatial scenes (Figure 1) and point out these actions characteristics.

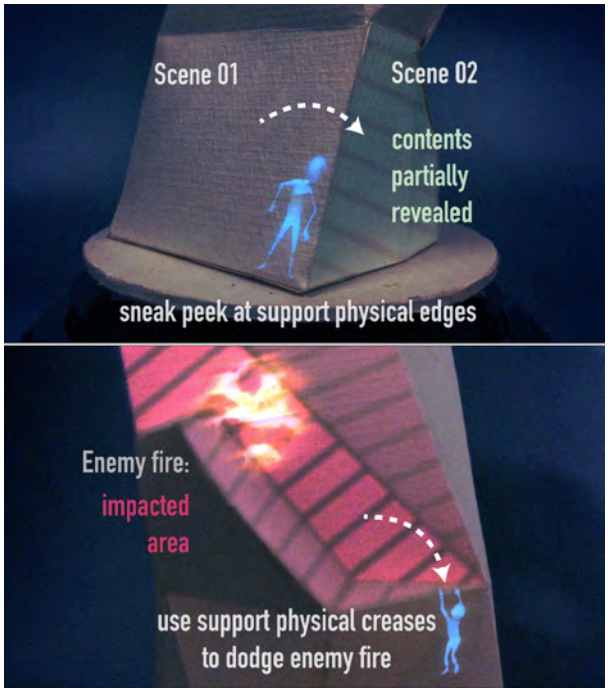

Figure 1 - Spatial actions are performed on physical creases mixing support object movement synchronised with animation designed to enhance the sense of depth in the projected scene.

Figure 2 - Dynamic SAR interactions designed for moving supports depend on image sensors to detect target object movements before updating contents to be projected. (a) Side by Side hand-held camera-projector device. (b) Reflective IR markers detected by cameras, used on Project Omote, (c) Scanning the environment with Microsoft Kinect sensor in RoomAlive project.

2 Related work

Most of the digital game experiences designed to have tangible interfaces using SAR are somehow based on tabletop game plays, in such way players can manipulate physical props like mini-figures or cards [5,6], having the projectors fixed above a table. As example, researches and commercial products such as Microsoft PlayAnywhere [7], and Lampix [8] seems to be aligned with this setup.

Other setups that allow projection to be expressed on 3-dimensional objects or surfaces with or without any degree of freedom are also available:

• ARmy [9] is a tabletop military strategy game where players can use primitive-shaped blocks to build up their own board before the game starts. A camera positioned above the board detects the blocks distribution to calculate the game dynamics. During the game play the board remains static while mini-figures are manipulated directly by players, thus updating the game instructions.

• RoomAlive [10] is a concept of immersive gameplay in which the player's room is 3-dimensionally scanned, allowing the system to make use of each physical surface to be part of a SAR experience game. During the game play the previously scanned surfaces remains static while the players' bodies movements update the game framework.

• SideBySide [11] proposes a portable device, which embeds an IR camera in a portable projector. Working similarly as a flashlight, which projects virtual information directly on planar objects. The player can freely move this hand-held device and project contents on apparently any dull surface. The contents are updated accordingly to the device orientation and readings from the IR camera (when using IR markers).

Projecting virtual contents onto moving objects requires the system to be aware of what content and how this content must be projected (Figure 2). The basic framework for this kind of applications relies on how the player's interaction over the target objected is detected by image sensors, and how fast and efficient the system reads data from these sensors to extract relevant information necessary to update the contents to be projected.

Other than limiting the target objects to be moved in a slow manner, running sophisticated computing vision subroutines in real time can overload graphic processors. This affects the application main routine, reducing interaction performance. Consequently this compromises the projection alignment with the target support, vanishing the illusion necessary to create convincing mixed reality. An alternative would require a extremely high-rate frame rate camera-projector module (above the average 60 fps) [12] connected to a system capable of running both computing vision subroutine and game play.

Instead, in our experiment we are working in the opposite direction, having the player to control the system which in its turn mechanically controls the physical object (Figure 3). This technique has the advantage of eliminating computing vision subroutines, thus having the processors dedicated to the generation of virtual imagery to be projected. Also, because the motor spins accordingly to known constants, it is possible to rotate it at any speed supported by the hardware (±10 to 100 RPM) without having the virtual projection to be misaligned with the physical object support.

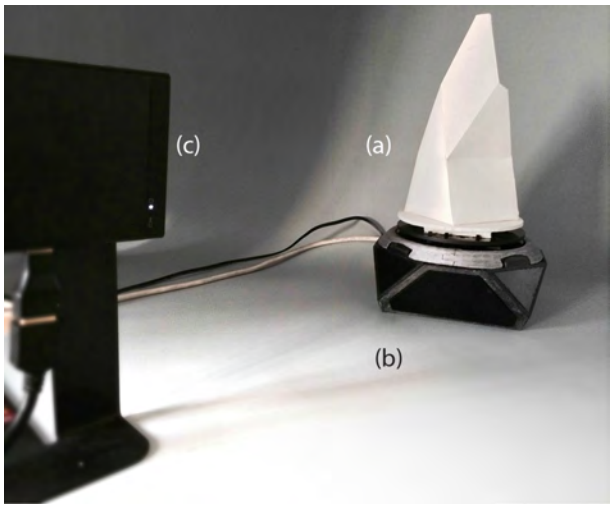

Figure 3 - An (a) physical object lays on top of (b) a controllable platform-hardware placed in front of a (c) free-focus laser projector. Projector and hardware connected to PC. Composition proposed as game console for this experiment.

3 Hardware and software development

3.1 Hardware

Originally, the physical object used as game stage can have as many degrees of freedom as the number of motors or actuators connected to the system. At this very stage, hardware and software are under development in order to facilitate its evaluation, while the main procedures runs satisfactorily. We

present the hardware including only one degree of freedom, as a turntable platform connected to a 200 steps/revolution stepper motor which is driven by a microcontroller subjected to PC commands. The system can recognise the angle of this turntable to project onto the support object accordingly. When hardware is turned on it will run a self-alignment procedure so the its rotor will start at origin.

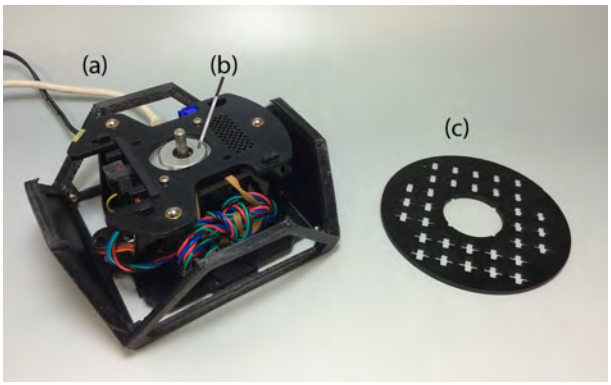

Figure 4 - Hardware is connected to a PC by (a) serial cable, which controls (b) a stepper motor. A support object is fixed on (c) a perforated disc mounted on top of the hardware.

The turntable platform is stabilised by 3 bearing balls located bellow its disc contact points. In addition, the platform disc is perforated in a pattern, which allows the posterior fixation of objects (Figure 4). Physical objects used as support for projection can be made of paper, or 3D printed, having its surface made dull for better illumination. Pico-projectors are portable devices designed to project contents during more casual situations. Their lenses field of view are not so wide and the brightness and contrast they offer are limited, since they use different source of light in comparison to desktop projectors, such as LED or Laser. The choice of using a laser projector is due to its focus free characteristic, perfect for the case where the distance between projector lens and projected surface isn't a constant. Also, the lenses characteristics of these projectors allows small details to be observed with satisfactory resolution on the nearly projected surfaces.

3.2 Software

Developed in Processing IDE [13], it consists of three basic modules, which are able to run a projector-calibration tool, an interface to the microcontroller, and the game play. Projector calibration is made every time a new support is changed or when projector or turntable changes their position in relation to each other. The motor control sends data to the hardware depending on game instructions. These data are degrees of rotation the motor is set to perform and revolution speed. Motor control can also read if hardware has completed its self-calibration procedure in order to setting up rotor at origin. The game play runs a basic physics class capable or allowing the player to move the game main character along the scenes. The game is composed by a sequence of planes, designed and mapped to each polygonal surface found on the physical object. When the player moves through the game, the motor is instructed to rotate accordingly. In this scenario, motor can run smoothly, following the character position, or revealing next scenes abruptly if this is the intention of the stage design.

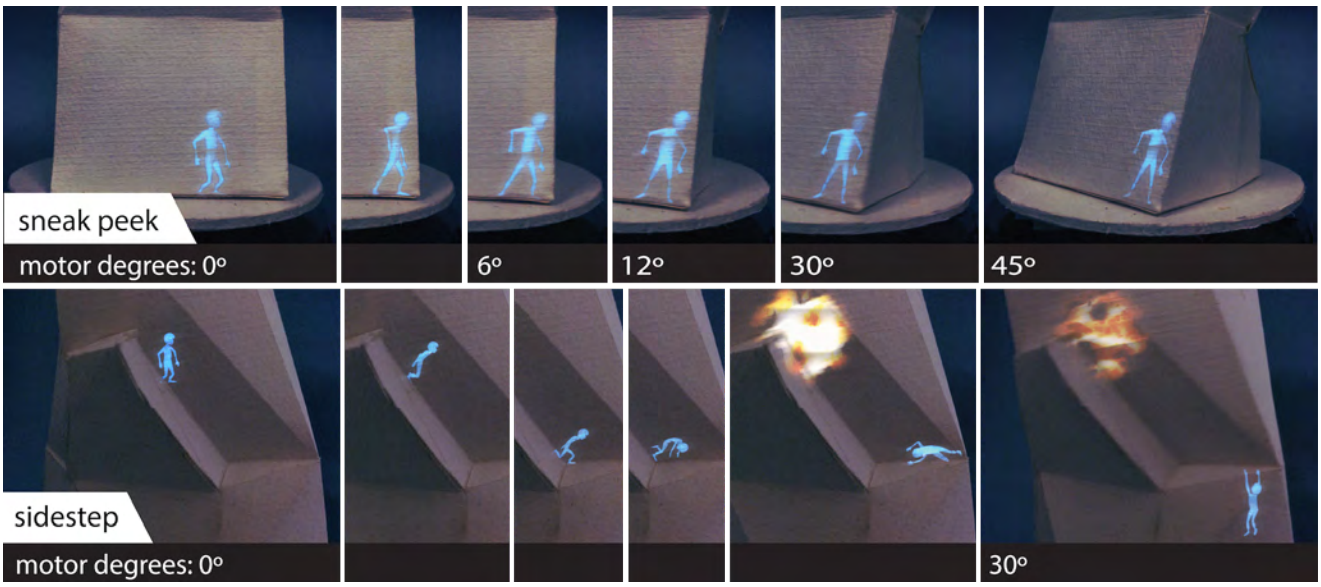

Figure 5 - Spatial actions designed for the turntable platform hardware. Animation plays aligned with the rotation of object support enhancing the illusion of depth.

4 Spatial actions

We suggest two spatial actions, which can be performed by the game main character in order to enhance the perception of 3D space or depth in SAR interactions. Spatial actions are designed to combine digital animation and physical object movement in such way players have the illusion of volume coming from the projected surfaces. These actions are provided by making use of the physical creases found on the support object and giving to it an opportunity to serve as attributes in the game such as places the game character specifically can use to perform spatial actions, as seen in Figure 5.

To follow with the explanation about spatial actions, we would like to highlight two spatial actions of them, which are used in our application: Sneak peek and Sidestep.

4.1 Sneak peek

Sneak peek actions take places on vertically oriented creases representing a wall corner from where the avatar can "spy" contents on its adjacent polygons in stealth mode. The crease angle between polygons must be physically constructed to make possible such kind of action inside the game. As a default value, if the crease angle between adjacent polygons are greater than 60°, the system will accept sneak peek action on this location. By performing this command, players can anticipate what is coming on next game scenes and get ready for a duel or reward, or even finding another path to go in order to avoid that area of the game.

4.2 Sidestep

Sidestep actions allows a runway manoeuvre to dodge enemy fire or seek temporary refuge. In a traditional 2D platform game, player would block or try to escape from enemy's fire while being trapped on a 2D coordinate system. In a first person shooter game, players can also dodge left or right from enemy's fire by hiding behind walls or door frames. In our game, we experiment a hybrid dimensional action, allowing a 2D character to make use of the physical geometry of the object onto where it is being projected. The sidestep action can be performed in spots where crease angle found on support object allows this kind of interaction between animation and support object. That is, when adjacent faces are not flat. Sidestep action is designed for vertical or horizontal creases, but subjected to the game design basic rules.

4.3 Other actions to be developed

There are other actions that can be applied to the game in order to explore the geometry and solid properties of the physical object. As example, the use of tunnels or holes, proposed on the game as means to transport the character from a place to another, can bring the discussion about how hollow the physical object can be, and how the use projection mapping can influence our perception toward the characteristics of the object as we know.

5 Projection and other technical aspects

5.1 Projector calibration

The projector calibration is a procedure used for solving the projector projection matrix. Specially for dynamic projection mapping, that is to say, when the spatial relationship between projector and illuminated object are constantly changing, this matrix is a key element for ensuring the precise alignment between projected virtual content and physical surfaces. The matrix contains information about the projector intrinsic and extrinsic parameters, such as the projector field of view, the projector screen size, the projector center position and orientation, among others. These parameters are stored in the form of a 4 x 3 matrix which are used to convert a 3D point in the virtual world to its correspondent 2D position on the projector projection plane. Thus, every fragment existing in our game application has its 3D coordinates multiplied by the projector projection matrix in order to display correctly on top of their correspondent planes in the physical support object.

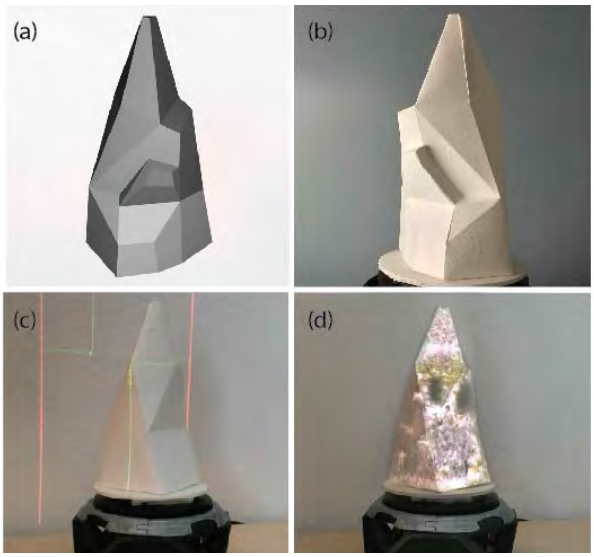

Figure 6 - a) 3D model representation of the physical object (b) used as base for projection; c) Crossed cursor projected on object for projector calibration setup; d) Virtual contents projected onto the physical object after calibration.

We have developed a tool in order to calibrate the projectors used in our application. The calibration process basically compares a physical object to its equivalent 3D virtual model (Figure 6 a and b). We have adopted a calibration method based on the single-camera calibration method suggested by Richard Radke [14]. This method requires the user to arbitrarily chose six non-coplanar vertices stored in a desired sequence. Having the physical object completely covered by the projection frame, the user can use a mouse cursor. to click over the previously determined vertices in the same sequence as those have been stored (Figure 6 c). Next, the system has enough information from the collected correspondent points to solve the projection matrix and project contents (Figure 6 d). The projector is said to be calibrated as long as the projector and physical object remains in place. Ideally, projector and the object platform should be fixed on the same base to avoid recalibration in case projector must be relocated.

5.2 Dynamic projection mapping

So far, the projection calibration can provide us relative accuracy for mapping physical objects in order to have them prepared for projections. However, similar results could be easily achieved using geometric distortion correction [15], that is, manually (e.g. using a mouse cursor) adjusting the image plane by dragging its corners until it gets visually aligned to the surface of the object one wants to project onto.

In order to have dynamic projection mapping we need to keep track of the physical object position and orientation after calibration process is performed so the system can automatically recalibrate the projector on the fly. As mentioned before, the current system developed for our game

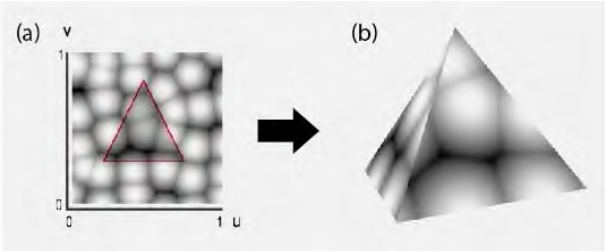

Figure 7- a) bitmap texture with a triangle representation of the 3D model (b) faces. b) 3D model having its faces textured by the selected portion of bitmap in (a).

application does not rely on any sensor, such as cameras, in order to track and retrieve information of the physical object being used as support for the projection. However, since the object is manipulated by user via system instructions, the system always knows where the physical object is positioned, preserving the calibration results as expected. A similar experiment [16] was conducted by the artist Cyril Diagne, having a physical object spinning at a controlled speed, which allowed the system to be aware of how to perform the projection mapping as expected.

5.3 Projection and game objects layers structure

In 3D applications, the process where users load bitmaps to be used as texture on the top of 3D models and set up how these planar images should match the geometry of the model is known as UV mapping (Figure 7). In such applications, U and V are the standard coordinates representing the axes X and Y within a face. Since 3D models can have very complex geometry, mapping each face individually to a portion of the bitmap texture can be very laborious. These applications usually provide tools to make the mapping process less complicated. By projecting the texture on the 3D geometry, users can associate many polygons to match the texture as desired. By choosing the right UV projection method, the modeller can have a good starting point to continue refining the texture on the model.

As with the UV mapping processes available on 3D applications, physically projecting 2D graphics on the surface of physical objects results in a certain level of distortions that should be handled somehow. The most well-known distortion related to projectors occurs when the projection surface is tilted in relation the projector itself, causing a rectangular image to display as a trapezoid. This scenario is almost inevitable for projection mapping setups where artists wants to project multiple images on complex objects having different angles and distances between their faces, requiring a equally complex technique to overcome this condition.

In the case of our application, bitmap texture used on the background, which are intended to be flat, can be projected within a certain amount of distortions using a basic setup where each face projects its correspondent portion of the bitmap multiplied by the projection matrix. However, the same cannot be applied to moving or interactive objects within the game. As seen in figure 8, projecting objects which are manipulated by player (e.g. characters) or perform some animation inside the game, will flatten with the background if we follow the same UV projection method used until now. This issue becomes more visible when projection takes place while the character or animated objects stands next to the edge between two adjacent planes, thus being projected on both planes at the same time (Figure 8 b and d).

We have opted to display these interactive objects in a foreground layer, using a different technic, so these objects can have their volume and lively characteristics preserved. In addition, the objects displayed in the foreground feels like detached from the background.

In order to make it possible to combine different projection methods to support mapping game objects, including collision surfaces such as floors, walls, and interactive characters or items, with the physical object geometry, we have developed a structure containing different layers, some hidden, for system and design evaluation, and some are rendered, as seen in Figure 9. For simplicity, we are presenting a 3D cube model as a physical support object to explain the structure of the layers and how they are setup individually.

Important to mention that, one of the main concepts of this project is to provide some sort of tangible game design interface, allowing a game to exist in multiple ways, slightly changing its complexity in accordance with the geometrical characteristics of the physical object the player decides to project the game onto. However, many of the processes discussed in the following subtopics, related to the game layer configuration, are setup manually, representing a limitation for using or designing complex geometry to be used in our project during this early development stage.

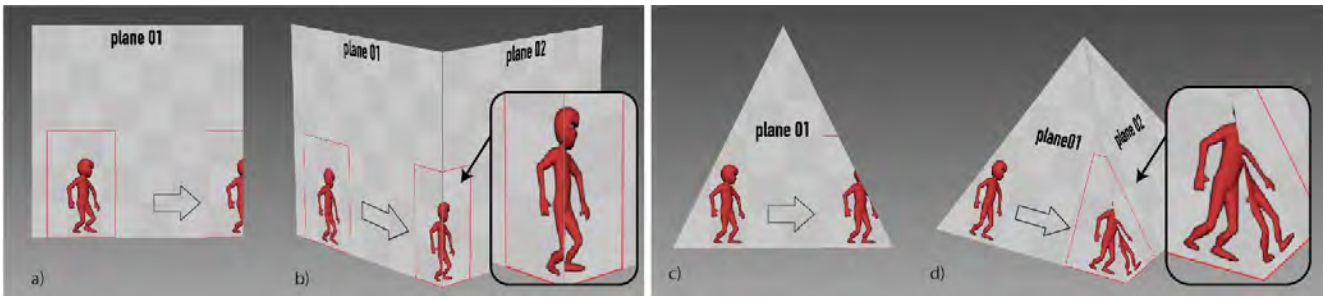

Figure 8 - a) Game character contained inside game map layer is moving to the edge its current plane. Seeing from the perspective of the player, character seems to be flattened against the plane, and its appearance becomes strangely distorted when the character is being projected on between two adjacent planes at the same time (b). The same problem is aggrieved when adjacent planes share edges non-perpendicular to the plane base (floor) (c and d).

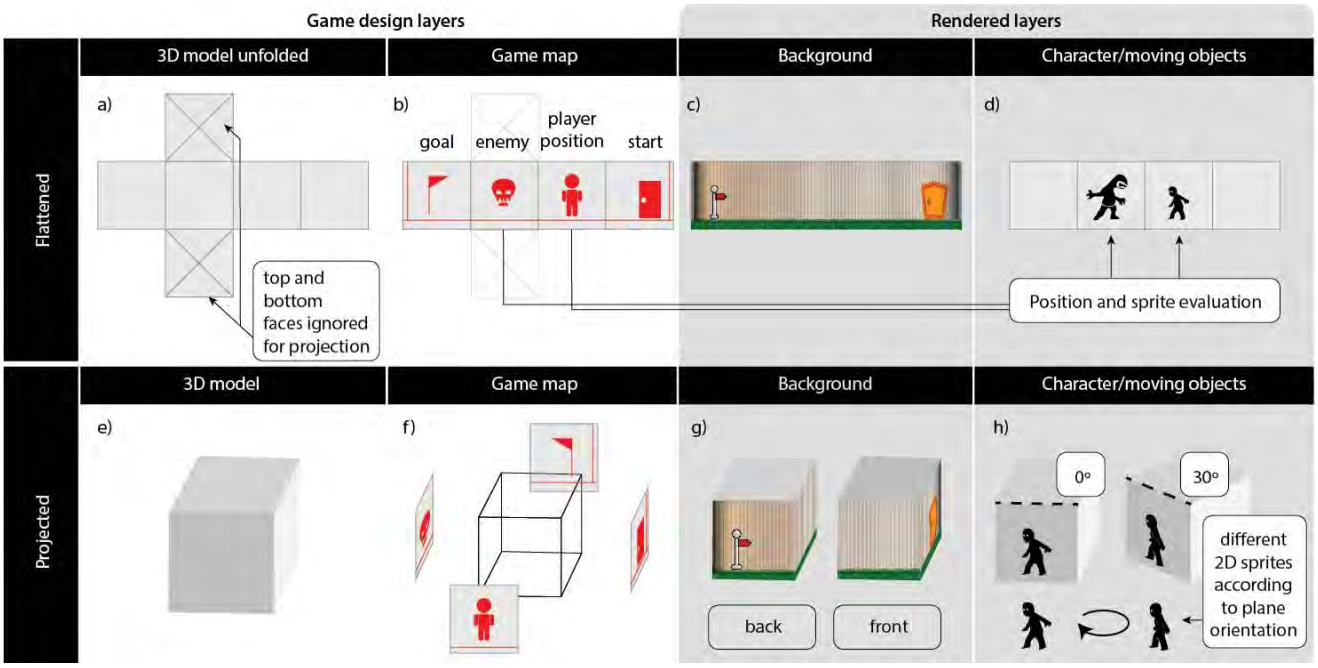

Figure 9- Game layers structure overview. a) 3D model unfolded, b) Game map (stage design) based on model unfolded layout, c) background texture bitmap, d) characters displaying on foreground after retrieving their position from game map, e) 3D model, representation of the physical object, f) Game map split and positioned according to 3D model planes, g) background texture applied/ projected on the 3D model, h) sprites display according to plane orientation in relation to the projector center.

5.3.1 Physical object and plane conversion

As explained earlier, 3D applications offers some tools used to facilitate the process of unfolding 3D geometry into a plane and mapping each polygon face to a portion of a bitmaps texture. Figure 10 - a shows a some possible way (Cylinder UV projection) of unfolding the 3D model used on our application. Projection unfolding methods don't take into account every single face of the model, but instead their relative position to the UV projection plane. All the faces visible from this plane are flatten, losing on dimension, resulting in distortions that might need some attention. A cylinder projection includes UV projection planes around the object and sew the view of each plane into one single plane that can be displayed on some UV editor window. On the other hand, the paper sheet unfolding method is not resulting of a projection analyses. Instead, this method maps each face to a plane individually, preserving their corners angles and their connection, whenever possible, to their adjacent faces. Also, all the edges are kept in proportion to each other (Figure 10 - b). Although the main purpose of using this unfolding method is to create a printable version of the 3D model, so it can be built as a physical object later, we have adopted this as a UV map layout to be used on converting the game map layer to the correspondent faces on the physical object because proportions remains undistorted. This way we can make sure that if a character on the game (virtual) moves 10 units to the left, the same applies to the physical model.

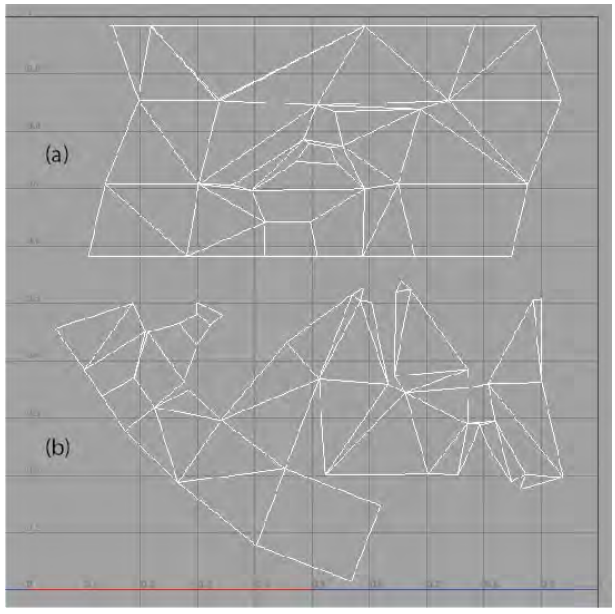

Figure 10 - a) Cylinder UV projection (generally provided by 3D applications) and b) the paper sheet unfolding method, used for building paper craft folding models.

5.3.2 Game map layer

This layer is used to design the game (stage) map, including collision surfaces which defines stage routes and goals. In addition, collectable items, tunnels and challenges spots for the game play can be decided on this layer. We have developed a tool which allows the creation of these game objects having the physical object planar map (defined in the previous step) used as a layout. See Figure 9 (b and f). By simply drawing lines and dragging them with the cursor, we can setup the basic collision surfaces like floors and walls for a stage design (Figure 11 - a). After the game map is decided, each fragment relative to a model plane is subsequently

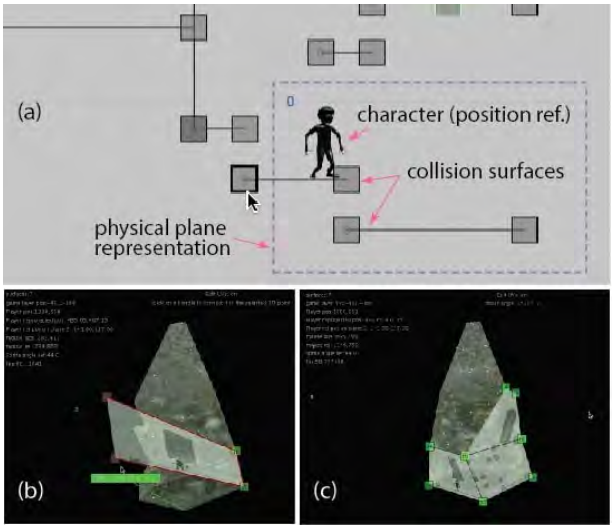

Figure 11 - a) Some game objects such as collision objects (floor, walls) are being designed inside the limits of the plane, represented by dashed lines. Every portion of the game map contained inside this plane representation becomes a game map fragment, later attached on the 3D model correspondent faces, as seen in (b) and (c).

connected to its 3D respective faces as seen in figure 11 (b and c). Because of this step, we can track the position of the moving objects, including the main character (manipulated by players) in relation to the 3D model center. This transform information is used later to render characters and other dynamic objects on top of the background layer, and will be detailed on the respective subtopic.

5.3.3 Background layer

The background layer is nothing but a bitmap texture designed to be mapped to the 3D object. Since the game map layer is not visible during the projection, the background texture is aimed to reinforce the game layout, including visual indications of floors, walls, holes and obstacles on their respective positions. Because of this dependency existing between the background and the game map layer, we have used the later as a reference to draw textures and other elements, on an external graphics editor application, and exported a bitmap image, used in our experiment.

This bitmap image is intended to be projected on the physical object the same way as it displays on the 3D model. Thus, after we have mapped the 3D model using traditional techniques for 3D texturing (on an external application), we were then able to retrieved the UV coordinates contained in each vertex of the 3D model and finally convert these coordinates to be used on the projector plane. As a result, we could project each 3D face to its correspondent surface on the physical model.

Currently, the background texture used in our application is flat and includes no bump (relief) nor other sublayer rendering instructions. We believe this texture layer can evolve to incorporate additional sublayers, such as normal or parallax mapping, causing a more interesting sense of perspective for the player during the game play.

5.3.4 Characters and moving/animated objects layer

This corresponds to the topmost layer where objects are animated or manipulated by user input commands. Objects contained in this layer render using a different technique than that used on the background layer. The reason is that having these “floating” objects dependent on the UV coordinate for each static face, would cause them to look flattened. In addition, when the animated object swipes from a plane to its adjacent plane it could be clipped out (Figure 8). A simple solution for this issue was to project the objects laying on this layer over the background separately. This way we make sure the object is always projected in alignment with players’ point of view (projector center). Let us take the example of the main character to broaden the scope of this technique.

All the actions (animation) performed by the main character are condensed in a single bitmap file as shown below, in Figure 12. This sprite sheet contains all frames used for the animation correspondent to each action of the main character.

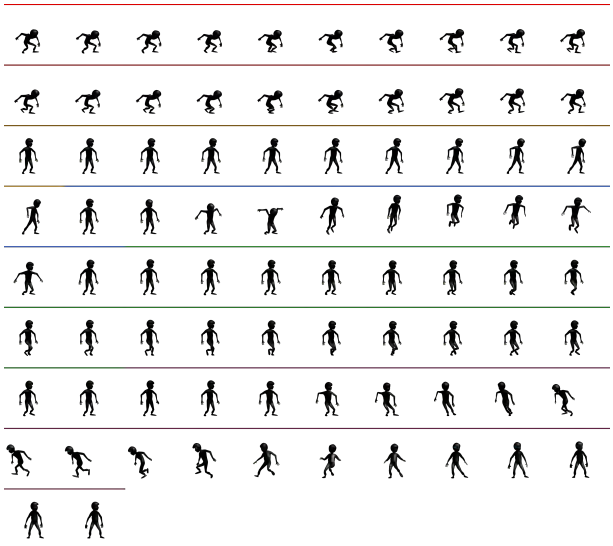

Figure 12 - Main character action sprite sheet for 0º plane orientation. Every coloured line represents an action performed by players during the game.

During the game play, the character is controlled by the player and its position is updated inside the game map layer, where character’s collisions against the game objects is evaluated. In order to extract the character position from the game map, we need to perform some transform operations. After identifying which plane is holding the character, we transform the 2D position of the character inside the game map to its 3D position on that specific plane - on the 3D model. We offset this point position to the 3D model center and finally multiply the result to the projector projection matrix in order to have the character correctly projected on the physical model.

We still need to be aware the projection is being performed at an angle, incurring in the so called keystone effect, which might cause the character to display more or less tilted. But because we have access to the orientation of each plane in the 3D model we are able correct this distortion by multiplying the character texture corners to the projector matrix, thus interpolate the result for each pixel within that area.

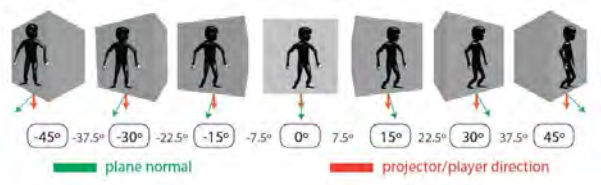

Figure 13 - same sprite frame from main character displayed according to a different sprite sheet set.

Finally, we add a sense of volume to the character and distance from the background by rendering the character according to the difference of the plane normal where character stands and the projector center direction, as observed in Figure 13. The red arrow represents a vector from the center of the plane to the projector/player direction, while the green arrow represents the face normal direction. We have prepared 13 sets of sprites contained all animation frames (see Figure 12), from different views: -45° to 45°, within intervals of 7.5°, corresponded to the motor minimum interval a game command will allow the motor to spin. The system will display the character's action from one of these sets, by considering the motor current angle, the projector center and the character position.

The projection method described in this subtopic intentionally distorts the object being projected to satisfy the player's unique point of view. Because of this arbitrariness, it is possible that some of the projected objects, including characters, might result in visual limitations (e.g. perspective distortions) for viewers situated somehow distant from the projector center.

6 Conclusion and future work

We have introduced spatial actions to be performed in digital games projected onto 3-dimensional moving support object with the goal of mixing this sort of interaction with tangible interface for games. Traditionally, game consoles embeds a CPU able to load data from external removable media and process it in real time according to the player's commands. The same game can be played, *"as is"*, in any compatible console. Instead, we propose the concept where the same game can vary infinitely according to the physical environment where it is projected onto. This concept of game can be developed bearing in mind to run a set of constant rules, such as the game objectives, challenges, story, characters, all previously defined by the game developer. At the same time, all these rules can be distorted by the physical surface onto where game is decided to be played. We envision a moment where this sort of hardware and software would be available for game designers and players so they could produce creative customised contents to be shared among other players. For the next steps, we are aiming to produce a more solid and standalone concept for hardware, including a self-calibration step, improving the game engine and game design tools, including procedural methods to build up the game layout randomly if desired and intuitive methods for 3D scan support objects.

Acknowledgements

We would like to thank Professor Kazuhiro Kato and Goshiro Yamamoto from Nara Institute of Science and Technology for point out the processes which made possible for us to build our customised calibration tool for the development environment used in this project.

References

[1] Milgram P., Takemura H., Utsumi A., and Kishino F., *Augmented Reality: A class of displays on the reality-virtuality continuum*, in Proc. SPIE 2351, Telemanipulator and Telepresence Technologies, 282, 1994.

[2] Nobumichi, Asai - *Omote / Real-time Face Tracking & Projection Mapping* - (https://vimeo.com/101225231), 2014.

[3] Sharp,Simon ; Jenkins, Tom - *Speed of Light / aka / The World's Tiniest Police Chase* (https://vimeo.com/43239312), 2012.

[4] B-Reel - *EELS 3D projection mapping multiplayer game* - (https://vimeo.com/31952864), 2011.

[5] *Do-It-Yourself: A Portable Digital Map for Tabletop Role Playing* - (http://www.gamergroup.com/page.roleplaying-game-articles.b.2827.r.1.html), 2010.

[6] Vanguard-gia (http://vanguard-gia.blog.jp/archives/1025942102.html), 2015.

[7] Wilson, Andrew D. *PlayAnywhere: A Compact Interactive Tabletop*, in UIST '05 Proceedings of the 18th annual ACM symposium on User interface software and technology, pp. 83-92, 2005.

[8] Lampix - http://lampix.co/

[9] Dolce A., Nasman J and Cutler B., *ARmy: A Study of Multi-User Interaction in Spatially Augmented Game*, in 2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, pp 43 - 50, 2012.

[10] Jones B., Sodhi R., Murdock M., Mehra R., Benko H., Wilson A., Ofek E., MacIntyre1 B., Raghuvanshi N. and Shapira L., *RoomAlive: magical experiences enabled by scalable, adaptive projector-camera units*, in UIST '14 Proceedings of the 27th annual ACM symposium on User interface software and technology, pp. 637-644, 2014.

[11] Willis K.D.D., Poupyrev I., Hudson S.E. and Mahler M., *SideBySide: Ad-hoc Multi-user Interaction with Handheld Projectors*, in UIST '11 Proceedings of the 24th annual ACM symposium on User interface software and technology, pp. 431-440, 2011

[12] Watanabe Y., Narita G., Tatsuno S., Yuasa T., Sumino K. and Ishikawa M.: *High-speed 8-bit Image Projector at 1,000 fps with 3 ms Delay*, in The International Display Workshops (IDW2015), (Shiga, Japan, 2015.12.11)/Proceedings, pp. 1064-1065, 2015.

[13] Processing IDE - (https://processing.org/overview/).

[14] Radke R.R., *Computer Vision for Visual Effects*, Cambridge University Press, pp 216 - 218, 2013.

[15] Geometric Distortions of the Image (http://wtlab.iis.u-tokyo.ac.jp/~wataru/lecture/rsgis/rsnote/cp9/cp9-3.htm).

[16] Mapamok with Arduino walkthrough (https://github.com/YCAMInterlab/ProCamToolkit/wiki/mapamok-with-Arduino-walkthrough).