Framing-Based Camera Tool for Artists in Game Development

Mathias K. Berthelsen
mkbe11@student.aau.dk

Gustav Dahl
gdahl11@student.aau.dk

Benjamin N. Overgaard
boverg11@student.aau.dk

Andreas M. Thomsen
amth11@student.aau.dk

ABSTRACT

Camera control in games is important, for example to create cinematic graphic effects or to guide the attention of the player. Unlike the camera in pre-rendered movies, a game camera should adapt to the player's movements. This paper describes the design of a tool that allows artists to frame the camera in games where the player moves along a pre-defined path. To this end, we used participatory design methods to understand how artists typically work with computer animations and found that they prefer to work with keyframing animation. This concept has been incorporated into the proposed Framing-based Camera Tool (FCT) in the form of framings. A framing consists of an influence point and a group of camera settings. Artists are able to define a group of framings by adjusting the position, orientation and field of view of the camera. Then, the game's camera interpolates between the framings automatically based on the player's position. Through an evaluation of FCT, we show that it allows artists to create and design dynamic camera animations.

General Terms

Game development

Keywords

Game development, camera system, interpolation, workflow, participatory design, collaboration

1. INTRODUCTION

There are many ways to design camera motions in games. Fundamentally, one can distinguish between *cinematic sequences* and *interactive gameplay*. These two are typically considered mutually exclusive, because cinematic sequences per definition are non-interactive [7]. However, it is possible to mix those two, such that the camera can dynamically adapt to game events and player input. This means that a camera sequence does not have to be viewed in exactly the same way every time.

This paper presents an approach to creating a tool for a specific group of artists whose main work domain is pre-rendered animation. Creating a tool for these artists is challenging since they have knowledge and experience working with movies, which is a pre-rendered medium, whereas games are dynamic and interactive. Our tool allows users to define camera movements in a game where the player character moves along pre-defined paths. It has been designed in cooperation with the artists using methods from participatory design. The main finding from the initial participatory exercises was that the artists are quite comfortable with using keyframing animations. Therefore, our tool is based on the concept of framings and was named *Framing-based Camera Tool*. Our evaluation of the proposed camera tool shows that it lets artists design and create dynamic camera animations.

The outline of the paper is as follows. We discuss related work in the next section. In Section 3, we describe how the artists contributed to the design of FCT. Section 4 goes through the design and implementation of FCT, while Section 5 shows how we evaluated the tool. We conclude with a summary and future work in Section 6.

2. RELATED WORK

Automatic camera control is an active research field. In 3D computer animation, a virtual environment is rendered in frames from a specific point of view. This is a complex process that involves technical and artistic tasks, typically requiring the work of several professionals for a long time period [5]. Developing a system that makes the camera animation automatic can be beneficial at both a professional level (to reduce cost) or at an amateur level (to increase quality). Systems that emphasize automatic behaviour have been proposed [4, 5]; these use an automatic approach for controlling a virtual camera driven by AI techniques. However, the focus of the camera tool proposed in this work is on artistic control. Hence, an automatic approach is not feasible.

In a behind-the-scenes documentary, the developers behind the PlayStation game series *God of War* talk about the importance of being able to frame the scene [16]. Depending on the context, it is important for them to frame the scene so players know where they should be heading next. They use camera zones to determine what area the player is in and then activate the corresponding camera for that zone. This example shows that it is important that the artists are

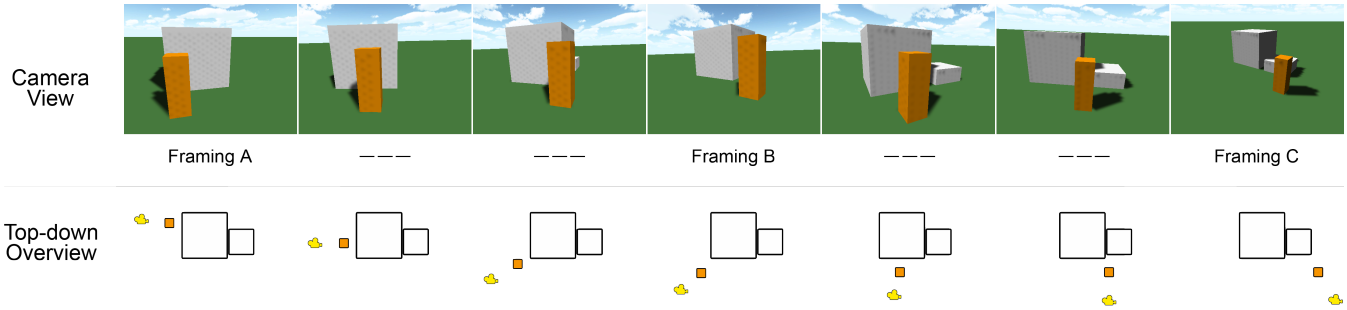

Figure 1: An example of a camera interpolation between three framings (A, B and C). The framings contain position, rotation and field of view. As the player character (orange cube) moves, the camera follows him by interpolating between the framings.

able to crate framings themselves.

The main requirement of a framing-based camera tool for games is that it can go from one camera setting (A) to another camera setting (B) depending on the player's position or certain events. For instance, A and B can be different in position, rotation and field of view. Figure 1 shows an example of how the camera interpolates between three positions.

The tool Camera Path Animator [6] can be used for creating animated cameras within the game engine Unity [15]. As the name suggests, it works by animating the camera along a specified path, which can have various shapes (e.g., Bezier and Hermite curves). The tool is primarily targeted towards creating cameras that move linearly along a set path, i.e., for use in cinematic sequences. It provides various ways of inserting, moving and deleting points, as well as changing settings such as field of view, speed, interpolation type and easing. Additionally, it has an event system for triggering certain events at certain points in the path. For our tool, it was important that the artists can create framings themselves and that the interface in general is designed for the artists; hence, we chose not to extend the Camera Path Animator, but instead took inspiration from it.

Participatory design has been defined as the participation of users in the design process of a system that is to be implemented in an organization [10]. By involving users in the design, their skills, experiences, and interests are taken into consideration, thereby increasing the likelihood that the system will be useful to them.

According to participatory design researchers, it is important that there is an active cooperation between the designer and the user [10]. This results in the designer gaining knowledge of the user's current work practices, and the user gaining knowledge of the technology being developed. It is also important that users take an active part in the analysis of needs, selection of technology, design and prototyping, as well as organizational implementation.

3. PARTICIPATORY DESIGN FINDINGS

The user group during the participatory design was a team of six third-year students from The Animation Workshop in Viborg, Denmark, who were developing a 3D point-and-click adventure game [9]. The team consisted of students from two different educations of the school: Character Animation and Computer Graphic Arts [8]. The students will be referred to as *the artists*. All the artists were familiar with making animation films with Autodesk Maya; however, it was the first time they tried to make a game. At the start of the project, the artists received an introductory course to the Unity game engine.

To create a tool that takes the artists' current skills and experience into consideration, thereby easing their learning curve, we used techniques from participatory design [3].

3.1 Activities

Four different activities using a combination of participatory design techniques were conducted with four artists from the team who had an interest in working with FCT. All activities were combined with open interviews as well as video recorded for further analysis. Each activity took 15-30 minutes per person.

The first activity, was an observation [3] of the artists working in Maya and Unity. The artists were asked to think-aloud [3] while performing simple tasks relevant to camera work in Maya and Unity. The purpose of this activity was to gain first-hand experience with the artists' current work practices in Maya as well as an understanding of how familiar they were with Unity.

Second, the artists were asked to list and sketch the features that they wanted the most FCT. This activity took inspiration from the freehand drawing technique [3]. The purpose of the activity was to get a foundation for which tasks the artists wanted to be able to do with the tool.

After sketching, the artists were asked to visualize the interface of the camera tool with a set of pre-made buttons and windows made of paper based on Unity's interface (see Figure 2). The activity took inspiration from the collage technique [3] and its purpose was to get a foundation for the interface design.

The final activity was made after the first design iteration of FCT. At this point, the features that were assessed as most important for the artists had been implemented; however, it was still necessary to receive input from the artists. Using

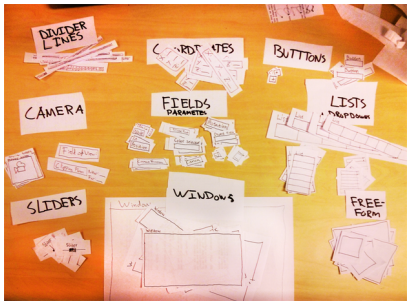

Figure 2: The collage activity. The camera settings and general interface were made out of paper.

the same techniques as in the initial observation, we had the artists perform simple tasks in Unity with FCT. From this, it was possible to see which changes should be made to the tool, especially regarding the interface.

3.2 Findings

From the previously-mentioned activities, we found that the artists in general wanted features that are similar to those found in Maya.

The artists deemed it essential that FCT should give them the freedom to frame the scene as they wanted, in order to focus on different elements in the scene.

An important feature that the artists used heavily in Maya when animating a camera in a scene is keyframing, i.e., the artists placed points on a timeline, and at each point adjust the camera according to how they wanted to frame the scene. During the initial observations, it was clear that many of the artists preferred the *Look Through Selected* [2] feature in Maya to adjust cameras.

When the artists needed control over how the camera should interpolate between keyframes, they used Maya's *Graph Editor* [1], which allowed them to adjust the interpolation for all camera settings using *animation curves* [1]. The artists often adjusted multiple curves at the same time and in the same window.

We also observed that all of the artists worked with two monitors and that they often spread Maya's interface out across both monitors to take better advantage of the space. They did this by creating separate windows for certain features.

A common request from all of the artists was the ability to quickly preview their changes. Instead of having to start the game and navigate the player character to a specific framing, the artists wanted to quickly jump to a specific framing to test out how it feels and looks.

Finally, we observed that the artists found Unity's *Flythrough Mode* [14]) particularly useful. This feature lets the player fly around with the scene camera as if they were playing a first-person game.

4. FRAMING-BASED CAMERA TOOL (FCT)

In this chapter, we describe the main elements of the tool that was developed with the artists. As mentioned in Section 3.2, the artists worked with two monitors; hence, the interface has been designed so they can drag elements of the interface to either monitor as they please.

4.1 Framing Concept

As mentioned in Section 3.2, keyframing is essential for the artists. In keyframe animation, each keyframe specifies a state at a specific point in time. Additional frames ("in-betweens") are interpolated between these keyframes.

In contrast to movies, games usually don't follow a linear structure; thus, keyframes for the camera animation can't be associated with points in time. In our case, the player is able to move freely on a pre-defined path. Therefore, FCT associates keyframes with the player's position in the game world instead of time.

To avoid confusion with traditional keyframing, the keyframes for the camera settings have been named *framings*. A framing consists of an *influence point* and a *camera* (see Figure 3). The camera holds all of the camera settings, such as position and rotation of the camera, as well as the field of view. The influence point only specifies a position. When the player's position is the same as an influence point, the associated cameras will determine the in-game camera. This means that the in-game camera will use the settings of this camera. Moving between influence points causes an interpolation between the associated camera. The interpolation can be customized in a graph editor as requested by the artists.

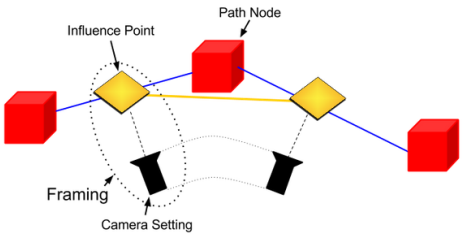

Figure 3: Illustration of the framing concept. A framing consists of an influence point and a camera.

4.2 Placing an Influence Point

In the current implementation of FCT, new influence points snap to pre-defined path segments, which are connections between path nodes. Each path segment keeps a list of associated influence points.

Influence points are represented by small diamond-shaped icons (see Figure 4). The connection between an influence point and a camera is indicated by a small line (see Figure 4). However, the influence point and the camera can be moved and adjusted independently.

Clicking on the path creates a framing. Creating a framing while having another framing selected connects the two. This connection is needed for manipulating the interpolation. As described in Section 3.2, graph editors are an essential tool for artists to adjust the interpolation between

keyframes. This has been implemented using the built-in graph editor in Unity (see Figure 5).

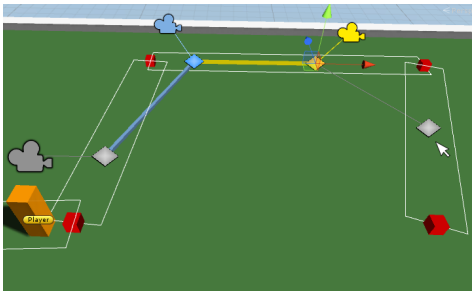

Figure 4: Influence points are placed by holding down the CTRL key and clicking with the mouse. They snap to the path defined by the path nodes (red cubes).

Since an influence point and its associated camera settings constitute a framing, selecting either one selects the framing; hence, both provide the same options in the settings window.

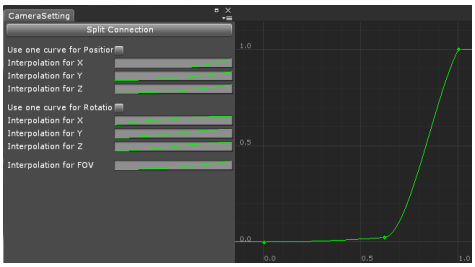

Figure 5: The graph editor can adjust the interpolation of the position, rotation and field of view.

4.3 Adjusting a Camera

As mentioned in Section 3.2, the artists preferred the *flythrough mode* in Unity. Besides the standard ways of editing cameras, FCT provides three additional ways; the first two can for example use the *flythrough mode*: *snapshot*, *be the camera* and *aim point*. *Snapshot* sets the active camera to the settings of the scene view camera. *Be the camera* enters a mode in which the active camera is always at the scene view camera until the artist exits the mode (see Figure 6). With the *aim point*, the active camera automatically looks in the direction of an aim point (see Figure 7). This *aim point* can be moved in the scene view like any other object.

4.4 Previewing the Interpolation

As mentioned in Section 3.2, previewing is essential for the artists' work, FCT provides three ways of previewing the interpolations. One way is to drag the player object around along the path via the *interactive preview*. A second way is to use the *slider preview*. Here, the artist can select a start framing and an end framing and then drag a slider back and forth, changing the interpolation accordingly. Additionally, a "play" button can be pressed to play the interpolation with the actual player velocity (see Figure 8).

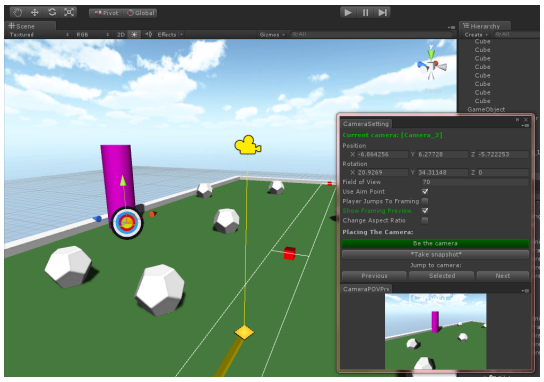

Figure 7: The aim point (located near the purple cylinder) allows for quick adjustments of the camera.

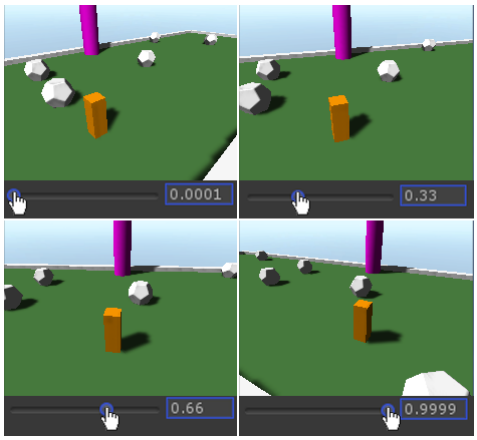

Figure 8: The slider provides a quick way to preview the interpolation.

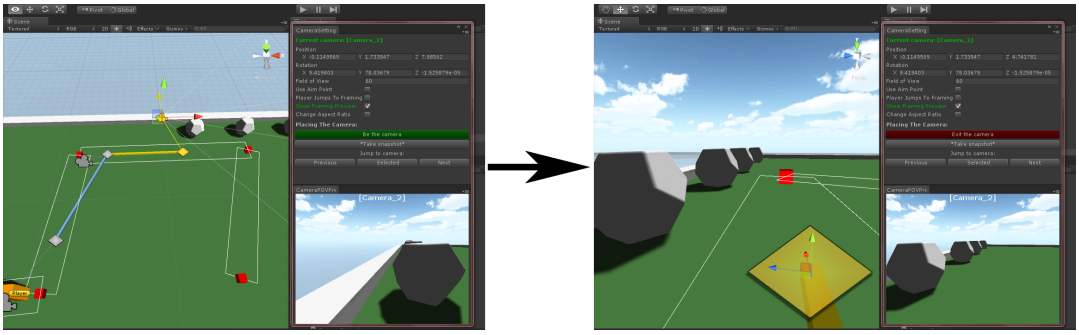

Figure 6: Pressing the green button puts the user in a special mode where the selected camera inherits position and orientation data from the scene camera. Notice that in the right image, the scene view and active camera have aligned and moved slightly. The preview image with the [Camera_2] label is how it will look in the actual game.

4.5 Interpolation Between Framings

Each influence point is placed on the player path. The interpolated camera settings (f_i) are a function of the player's position P_{pos}, which is used to interpolate between the camera settings c_1 and c_2 of the two influence points on the path closest to the player's position in both directions, with weights w_1 and w_2 based on the positions of the influence points.

$$f_i = f(P_{pos}) = w_1 c_1 + w_2 c_2 \quad (1)$$

The two closest influence points are found by a linear search from the player's position along the path segments in both directions.

The weights w_1 and w_2 are then computed from the distances d_1 and d_2 travelled in both direction until encountering an influence point.

$$w_1 = d_1/(d_1 + d_2) \quad (2)$$

$$w_2 = 1 - w_1 \quad (3)$$

If there is no influence point in one direction, the distance in that direction is zero. The artist is further able to manipulate the weights w_1 and w_2 for each camera setting through the graph editor. Evaluating a curve at w_1 produces new weights for these camera settings that are associated with the camera settings.

4.5.1 Path-Based Slider-Preview

For the slider preview, a player position P_{pos} is computed from the given slider value w_s (from 0 to 1) and the influence points of the start and end framing.

Drawing from the same logic as in the interpolation, a linear search is performed from the influence point of the start framing following the path in both directions, logging the total distance travelled until the influence point of the end framing is encountered. Using the total distance between the start and end framing, the interval [0,1] of the slider value is mapped to the sequence of the encountered influence points between the start and end framing. For slider values mapped to positions between two influence points, the position P_{pos} is computed using linear interpolation between these influence points.

5. EVALUATION

After designing and implementing FCT, we evaluated whether it allows to successfully perform the tasks that it was originally designed for (described in Section 3).

5.1 Method

5.1.1 Participants

Two groups of participants took part in the evaluation: three artists from the participatory design group and three artists outside the participatory design group. Both groups were students from The Animation Workshop [8]. Since the first group was involved in designing FCT, they had previous knowledge of the tool. The three other artists knew nothing about FCT and were used as a control group to avoid this bias.

5.1.2 Procedure

Each participant was evaluated individually (see Figure 10). Besides the participant, a facilitator and an observer were also present. The evaluation was split into seven parts:

1. Introduction, Consent and Demographic Questionnaire
2. Trying the Camera Path Animator
3. Basic Navigation in Unity
4. Training with FCT
5. Tasks with FCT
6. Creative use of FCT
7. Evaluation Questionnaire

During parts 3-6, the participants and their monitors were recorded. The procedure was the same for all participants. The evaluation sessions lasted for approximately 40-50 minutes each.

Introduction, Consent and Demographic

The participants were given a brief introduction and presented a consent form to allow video recording. The participants started by answering a short demographic questionnaire.

Trying the Camera Path Animator

The participants were introduced to a small demo of Camera Path Animator [6] (see Section 2). They were told to

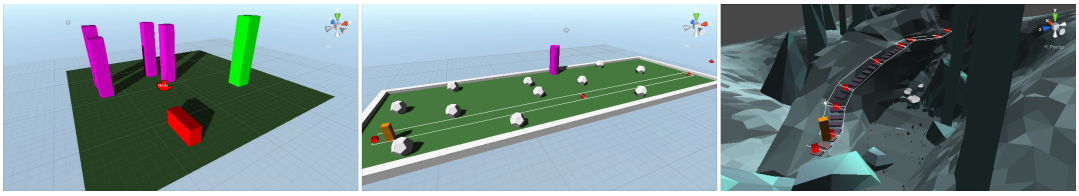

Figure 9: The evaluation consisted of three parts, each with their own scene. The first scene was used to learn about basic navigation in Unity. The second scene was used for the participants to try FCT. The third scene was for the creative task where the participants created their own framings.

Figure 10: The participants used two monitors while working with FCT.

move the player character around and notice how the camera behaved accordingly. This was to give the participants a context for the evaluation of FCT, as well as introduce them to the concept of camera movements in an interactive environment.

Basic Navigation in Unity
To ensure that all participants had a basic understanding of how to navigate in Unity, the facilitator gave a brief oral introduction to the basic functionality of Unity. This included a short description of the essential windows in Unity (scene view, game view, hierarchy and the inspector), as well as how to move and rotate objects. The participants were instructed on how to move the scene view camera around in Unity. They were also told about Unity's *flythrough Mode* [14]. To ensure that they actually understood how to navigate the cameras, they were asked to move the camera to three specific locations in the scene. This phase lasted 2-5 minutes.

Training with FCT
Before working with FCT, the facilitator gave a short oral introduction of the framing concept by referring to a printout of Figure 3. Afterwards, the participants were asked to open FCT. Here, the facilitator went through all of the major features in a semi-structured way by explaining each feature, one at a time, and having the participants try each one.

A path was already defined in the scene (see Figure 9). The participants learned about how to place and adjust the framings. They learned how to create and move influence points. They then went through features such as changing the position and rotation of a camera using the different methods; i.e., changing the values by hand; using the *be the camera* feature; using the *snapshot* feature; and using the *aim point*. In order to preview their changes, the participants were shown the *interactive preview* and *slider preview*. Afterwards, they were introduced to the graph editor.

This phase lasted 15-20 minutes.

Tasks with FCT
After the participants had tried out all of the major features in FCT, they were handed a piece of paper with five tasks. The purpose of this phase was to check if the basic functionality of FCT worked, based on whether the participants could complete the tasks, and observing how difficult or easy the tasks and functionalities were for the participants. The tasks were:

1. Make the camera's field of view change.
2. Make the camera tilt upwards.
3. Make the camera look at the tall pink cylinder.
4. Make the camera go from a low perspective to a bird's-eye view.
5. Change the interpolation of one of the previous assignments by changing the animation curve.

These five points reflect common features and tasks that are often used when setting up a framing and a camera interpolation. The participants had to solve the tasks themselves; the facilitator only intervened when the participant struggled with something, asked a question, or if unforeseen errors occurred with the tool.

This phase lasted 15-20 minutes.

Creative use of FCT
The participants were introduced to a scene with a modelled environment. They were then tasked to envision and sketch two ways for the camera to move as the player character moved through this environment. After drawing their two sketches, they were asked to implement both of these using FCT. The facilitator remained as neutral as possible for this part of the evaluation, but still intervened if the participant was struggling or encountered errors in the tool. This semi-structured approach could potentially illustrate shortcomings and missing functionalities of FCT.

This phase lasted 5-10 minutes.

Evaluation Questionnaire
The participants answered a short questionnaire about their experience with FCT.

Figure 11: The participants drew sketches of camera movements in the creative phase during the evaluation. There were no restrictions on how to do this; but most participants drew traditional storyboards. This figure shows two different participants' sketches and implementations in FCT. Participant A was part of the participatory design group and had 4-6 years of Unity experience, while participant B was outside the participatory design group and had less then six months Unity experience.

5.1.3 Materials

The Camera Path Animator [6] demo uses the "Angry Bots" game provided by Unity [13]. It shows how the in-game camera reacts to the position of the player.

Three scenes were constructed for the evaluation of FCT, each with gradually more complex geometry (see Figure 9). The first scene consists of simple boxes of various shapes and colors. The second scene is a simple sandbox-like environment with a pre-defined path. Small rocks have been added to make it easier to orientate oneself. The third scene depicts a mountainous environment with a staircase attached to the mountainside, again with a pre-defined path. Contrary to the two previous scenes, this scene has variety in both height and depth.

Two monitors were recorded using Open Broadcast Software [11] and a webcam recorded the participants' faces, together with the audio.

5.2 Evaluation Questionnaire

The participants had varied experience with Maya, ranging from six months to six years. Their experience with Unity was less than a year, with a single exception of a participant having 4-6 years of Unity experience.

The participants rated their agreement with a list of statements using a 5-point Likert scale ranging from "strongly disagree" to "strongly agree".

When asked if the participants *"felt empowered using the tool"*, four participants *agreed*, while the remaining two *strongly agreed*. When asked if they *"felt restricted using the tool"*, five participants *disagreed* while the remaining participant *strongly disagreed*. When asked if they felt that they *"got the tasks done quickly"*, four participants *agreed* and two *strongly agreed*. To find out how effective the participants felt using FCT, we asked them if they *"felt the tool allowed them to complete the tasks well"*. Three participants *agreed* and three *strongly agreed*.

Furthermore, they were asked whether they preferred the *be the camera* or the *snapshot* feature for adjusting cameras. Four chose *be the camera* and two chose *snapshot*. They were also asked how useful they thought the preview features were. The response choices were *not useful*, *somewhat useful*, *very useful*, *didn't use any of them* and *don't know*. Five participants found the preview features *very useful*, and one found them *somewhat useful*. Finally, the participants were asked what their overall favorite feature was. Three participants noted the *be the camera* feature as their overall favourite feature, while the *slider preview* was the favourite for the other three participants.

To see the full demographic and evaluation questionnaire, see the attached worksheets.

5.3 Discussion

Due to several factors, the results gathered from the evaluation are not sufficient to draw statistically significant conclusions. One of these factors is the low sample size of six participants. Other factors include an artificial setting where the participants had an unrealistic time to both *learn* and be able to *use* FCT. Besides this, they also had to learn the fundamentals of Unity, all within a time period of one hour. Some of the participants believed they would be able to learn the tool properly if they had a little more time available. Ideally, the artists should have enough time to become "advanced users" before the evaluation even began. This would ensure that the evaluation would look into FCT's functionality instead of its usability.

One interesting note is that the participants used some of the features in unexpected ways. For instance, one participant used the *aim point* as a pivot point to orbit the camera around. The feature was not designed with this in mind, but it turned out to work rather well. This, together with other instances, indicated that the participants had an interest in learning and using FCT.

Even though the participants appeared to be able to create the framings that they sketched in the creative phase, it should be noted that they at this point already knew about FCT's strengths and weaknesses. This potentially influenced what they sketched.

When comparing participants' sketches to their implementation (see Figure 11), they appear very similar. Furthermore,

whether or not the participants belong to the participatory design group seems to have little influence on their framings. For instance, Figure 11 shows sketches and framings created by one participant from the participatory design group and one participant who was not from the participatory design group. Additionally, the amount of experience in Unity seems to have little impact on the participants' framings since one of the participants in Figure 11 had 4-6 years of experience in Unity, while the other had less than 6 months of experience in Unity.

During the evaluation, one participant expressed that they would be satisfied with fewer options, e.g., that it was unnecessary to have both the *be the camera* and the *snapshot* feature. The same goes for the multiple ways of previewing. The participant found that the amount of options were maybe too high and cluttered the interface. However, in total, all features were used by the different participants; hence, it would require further investigation to find out what features were redundant.

Many of the features designed for FCT are based around similar concepts in Maya. Therefore, the usability of FCT may have given confounding results since all participants have prior experience in Maya. FCT was not tested on participants with no prior Maya experience.

6. SUMMARY AND FUTURE WORK

We have defined the basic requirements needed for a framing-based camera tool built specifically for a game where the player character's movement is restricted to a path. Based on our evaluation, the proposed camera tool lets the artist design and create dynamic camera animations. The framing-based camera tool proved versatile enough to accommodate creative freedom and even usages not originally envisioned.

It is difficult to compare the gathered data from the different phases in training, tasks and creative work. Since all of the phases had different purposes and time spans, it doesn't make sense to compare them against each other. Two suggestions to improve the evaluation of what is possible to achieve with FCT are: a) Let the participants recreate camera movements from other existing games; b) Let the participants create camera movements inside Maya and then re-create the same movements using FCT.

A limitation of the evaluation was the study of creativity, i.e., how the artists were able to get an idea, sketch it out on paper and then implement it with FCT. It raises the question on how to judge creativity and quality. One approach is to let professionals in the field of camera animation judge the end results of the framings anonymously [12]. We encourage more research into this field.

7. ACKNOWLEDGEMENTS

We would like to thank the artists that helped designing the Framing-Based Camera Tool. We would also like to thank Anja Perl and Emil Kjæhr from The Animation Workshop in Viborg for making this collaboration possible.

8. REFERENCES

[1] Autodesk Maya Online Help. Graph Editor. *http://download.autodesk.com/us/maya/2009help/index.html?url=Keyframes_and_the_Graph_Editor_Using_the_Graph_Editor.htm,topicNumber=d0e12559*.

[2] Autodesk Maya Online Help. Look Through Selected. *http://download.autodesk.com/global/docs/maya2013/en_us/files/Basics_Menus_Panels_Look_Through_Selected.htm*.

[3] K. Boedker, F. Kensing, and J. Simonsen. *Participatory IT Design - designing for business and workplace realities*. Viva Books Private Limited, 2006.

[4] O. Bourne, A. Sattar, and S. Goodwin. A constraint-based autonomous 3d camera system. *Constraints*, 13(1-2):180–205, 2008.

[5] P. Burelli and M. Preuss. Automatic camera control: a dynamic multi-objective perspective. 2014.

[6] Camera Path Animator. Animate Cutscenes in Unity with Ease. *http://support.jasperstocker.com/camera-path-animator*.

[7] M. Haigh-Hutchinson. *Real-Time Cameras*. CRC Press, 2009.

[8] http://www.animwork.dk. Bachelor's degree programs at The Animation Workshop. *http://www.animwork.dk/en/degree_courses.asp*.

[9] http://www.mobygames.com. Adventure. *http://www.mobygames.com/genre/sheet/adventure/*.

[10] F. Kensing and J. Blomberg. Participatory design: Issues and concerns. *Computer Supported Cooperative Work (CSCW)*, 7:167–185, 1998.

[11] Obsproject.com. Open broadcaster software. *https://obsproject.com/*.

[12] I. Sadeghi, H. Pritchett, H. W. Jensen, and R. Tamstorf. An artist friendly hair shading system. *ACM Transactions on Graphics*, 29(4):1, 2010.

[13] Unity. Angry bots. *http://unity3d.com/showcase/live-demos#angrybots*.

[14] Unity. Scene view navigation. *http://docs.unity3d.com/Manual/SceneViewNavigation.html*.

[15] Unity. Unity game engine. *http://unity3d.com/*.

[16] YouTube.com. God of War 3 Bonus Features - Camera Design in HD. *http://youtu.be/d13iQUmpXG0*.