Eurographics Workshop on Intelligent Cinematography and Editing (2022)
R. Ronfard and H.-Y. Wu (Editors)

Real-Time Music-Driven Movie Design Framework

S. Hofmann[1], M. Seeger[2], Prof. H. Rogge-Pott[2], Prof. Dr. S. von Mammen[1]

[1] Julius-Maximilians University Würzburg
[2] University of Applied Sciences Würzburg-Schweinfurt

Abstract

Cutting to music is a widely used stylistic device in film making. The usual process involves an editor manually adjusting the movie's sequences contingent upon beat or other musical features. But with today's movie productions starting to leverage real-time systems, manual effort can be reduced. Automatic cameras can make decisions on their own according to pre-defined rules, even in real time. In this paper, we present an approach to automatically create a music video. We have realised its implementation as a coding framework integrating with the fmod api and Unreal Engine 4. The framework provides the means to analyze a music stream at runtime and to translate the extracted features into an animation story line, supported by cinematic cutting. We demonstrate its workings by means of an instance of an artistic, music-driven movie.

CCS Concepts

• ***Computer systems organization*** → *Real-time operating systems;* • ***Applied computing*** → *Sound and music computing; Media arts;*

1. INTRODUCTION

Game engines are no longer just about games. Recently, the popular *Unreal Engine* caught everybody's attention when its owning company Epic released a tech demo on the engine's most recent features. The associated short film 'The Matrix Awakens: An Unreal Engine 5 Experience' impressively underlines the capabilities of game engines in making movies. The interest in this artistic usage of game engines already emerged when the television series 'The Mandalorian' hit big in 2019. By replacing common green screens with LED-walls and the use of real-time computations, the production process has been augmented in many ways. First, actors are much more immersed throughout the recording due to the live projection of the virtual surroundings on these LED-walls. Auxiliary to that, the shooting can already be finalized without the need to add the background later on. But creating cinematic shots in game engines not only comes in handy as support for films starring human actors. It also expands the possibility of creating films such as 3D animations, VR 360° films and interactive movies.

To explore novel opportunities of game engines to support music movie production, we created an automatic music-driven cut framework for the game engine *Unreal Engine 4* (UE4) [Gam98], that integrates music analysis by means of the *fmod* api. The main motivation was to create a video in real time, where every action, look and cut of the scene is completely guided by a continuously read music stream. Thus, our approach translates the musical input to an animation story line, evolving the intensity of changes, i.e. basic transformations such as translation, rotation and scaling of objects and also their projections, boosted by cinematic cutting. Cutting to music is a rather common technique, but mostly the cuts have to be selected manually by a human editor. Thus, our system automatically takes care of finding the specific points in time when the beat drops, and of selecting the most interesting scenes and views from the entire video material. At each run, a new result is produced, and the element of surprise is part of the desired aesthetics.

We identified several, partially rather elaborate preceding works that we could use to drive the approach that we present in this paper. For example, there are sophisticated solutions for automatic camera positioning or automated cutting. We will detail related works in Section 2. But in order to arrive at an autonomous system such as the one we envisioned, we realised that we would have to engineer a novel technical solution. Our proposed solution delegates the cinematic control to the music while sticking to common practice guidelines of framing the associated actions in the scene with appropriate cuts and angles.

Our system can be divided into three parts: (1) The audio analysis component, (2) the cut system and (3) the view system. The first provides the information that drives the other two, based on a set of established cutting operations and manually provided *Focal Points* that capture interesting objects or views in the scene. An overview of the entire system can be seen in figure 1. It shows the flow of data that drives the system: Musical features of the sound file, depicted on the left-hand side of the diagram (border with double lines), are calculated in the first component. They determine the

DOI: 10.2312/wiced.20221052

behaviour of the cut system (bottom, border with dashed line) that makes the view (top, border with dotted line) transition to a new favourable state as well as controls the behavior of the objects in the scene. We will dive deeper into the explanation of the involved functionalities in section 3.

At each frame, a currently played sound file is analyzed with respect to the beat, frequency and amplitude. Each time a beat drops, the cut system selects a different cutting operation depending on a random component, the previous cut and the consideration of cinematic principles. Next, a new camera angle is determined at random. Lastly, the respective camera of a *Focal Points* can either be fixed in its position or move along a pre-defined path as a dolly shot, which is randomly decided as well. The view system then chooses a new view into the scene based on the greatest level of activity and cuts as directed by the cut system. To determine the level of activity, we implemented a concept we call *Focal Points*, which is a combination of a camera with a specific view on some objects, whose movement and other changes are also controlled by the output of the audio analysis. An example setup of such a *Focal Point* can be seen in figure 2.

The following section 2 explores related work. In section 3, we describe the functionality of our approach, with details on its prerequisites, the audio analysis, the cut system, and the view system. A short description of a demo instance is given in section 4. We discuss the results in section 5 and conclude with some possible future enhancements in section 6.

2. RELATED WORK

Generally, our approach falls in the category of automated camera systems, which have a wide range of application. They support, for instance, real life physical systems such as sporting events, medical procedures such as surgical simulations, or virtual cinematographic productions, e.g. when producing a movie out of game play. Hence, this is an extensively explored topic, but little research has been done when music plays the determining role and camera-captured visuals have to follow, and all in real time. The concept of realizing this is bipartite. On the one hand, there is the technical component, that we summarize under camera systems responsible for the timing of cuts and the positioning of the cameras in the scene. On the other hand, there is the design guideline, which is in charge of the cutting style and the selection of views and therefore controls the action that is visible in the final cut sequence.

2.1. MACHINIMA

The first-person shooter game *Quake* [Int96] lead to the production of early cinematic movies created from game play already in 1996 [Low08]. The then upcoming trend in observing other players engage in a match brought up the demand for a better view of the action from different perspectives. But as the viewers asked for a live broadcast, it was not acceptable to edit recorded footage afterwards. Further, a simple look over someone's shoulder was not enough, diversified sights were required [Low08]. The principle of *Machinima*, a combination of 'machine' and 'cinema' was born [PJSH13]. Although initially only concerned with better watching of online games, it quickly turned to a more general concept of developing cinematic sequences in game engines [MY09]. The main challenge in doing so was to create visually appealing footage and a narrative from scripted attributes such as dialogue, movements and gestures [ER07]. Thus, it connects video games with some forms of art [Pic06].

Over the years, four types of *Machinima* emerged [BBB*06]. The first category comprises the purest form, being in fact independent from game engines. These custom-built platforms are particularly strong in the field of camera control, the look of specific shots and lighting. Unfavorably, they are often limited due to missing real time computations. An example software of this classification would be *Microsoft's Virtual Stage* [Tho00]. A second *Machinima* group is formed by hybrid games. These involve filming mechanisms built into an interface to simplify the final production. Most of these applications include an editor, which makes it convenient to work with. Although sometimes there are issues with uneven libraries, there are various famous games built this way like *The Movies* [Stu05], *The Sims 2* [Max04] and *Second Life* [Lab03]. Problems with additional libraries are also a drawback of pure games, the third category. While they do not actually provide built-in mechanisms for *Machinima* production, it is still possible to use the technology, as can be seen in *Halo 2* [Stu04] or *World of Warcraft* [Ent04]. The last class is made up of modded games. Games such as *Half-Life 2* or the *Neverwinter Nights' Aurora Toolset* [Bio02] demonstrate the feasibility of creating *Machinima* by using and modifying a game engine's source code to fit the system to the movie's requirements [BBB*06]. By now, there are not only various *Machinima* game play movies, but also plenty of well-established programs and frameworks to create them [HZ12]. An exemplary system would be *Zuzen* by Munilla and Young, a cloud-based service, that uses the UE4 to produce cinematic videos [MY09]. To start the process, it requires a prespecification of the story and camera activity. First, the pre-defined plan is converted into a set of parameterized function calls, which are represented as so-called action classes. The execution of these classes is done within the engine. The so rendered movie is then recorded and can be downloaded via a http service. *Zuzen* further relies on the application *Darshak* [UB07], an intelligent camera planning system, which makes it possible to create individual camera directives. The overall mechanism is controlled by a directed acyclic graph [MY09]. Another cinematography software specially developed for *Machinima* is *CAMBOT* by Elson and Riedl [ER07]. Their application replicates the film making process by reading from a script with information on shot compositions, time-indexed dialogue and gesture commands. This script gets divided into consecutive scenes, that compose the final movie, supported by cinematic knowledge from a third party library. The developer has the additional option of defining constraints such as about the location, blocking and view of objects in the scene [ER07].

2.2. CAMERA SYSTEMS

Automatic camera systems are mostly realized by optimization algorithms that focus on camera path planning, occlusion culling and parametric specification, taking cinematic rules into account.

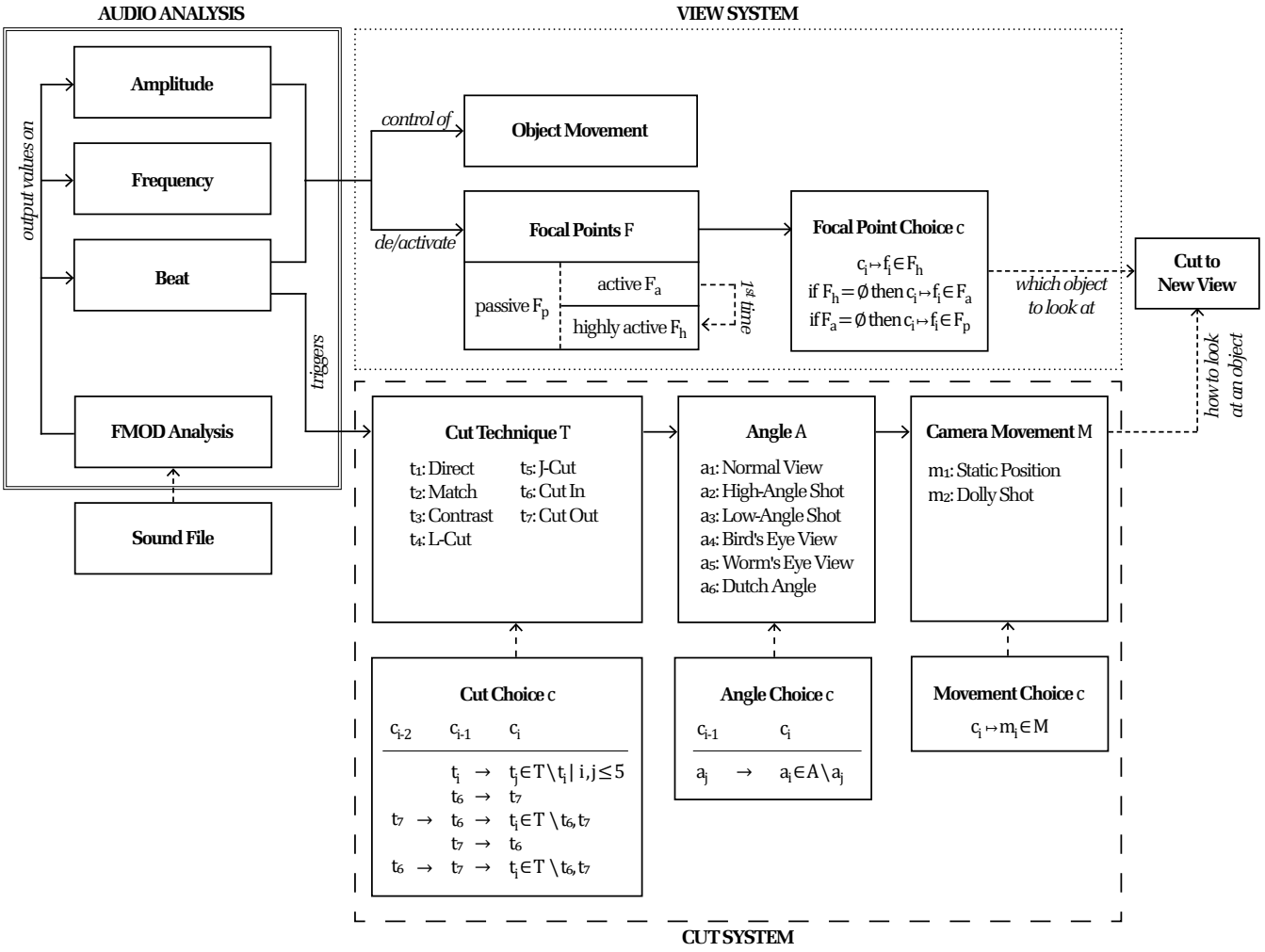

Figure 1: *Overview of the framework.*

2.2.1. Camera path planning

As early as 1994, Drucker and Zeltzer designed the walk-through of a virtual museum with intelligent camera control [DZ94]. Their system is based on analyzing the tasks required in a specific environment. With information about orientation, navigation and even unknown areas, the system prepares the presentation of the scenery to suit the external observers. The automatic positioning of the camera is based on an A*-search algorithm.

A more recent method comes from Meeder, who built the *Aesthetic Camera* to achieve virtual camera control in the context of e-sports in real time [Mee20]. His approach integrates the five rules of cinematography by Joseph Mascelli, namely paying heed to camera angles, continuity, adequate cutting, close-ups and a fitting composition [Mas65]. A cost function revolving around the geometry of the scene provides the basis for choosing the best camera location. The optimization is performed by the *Ceres Solver* for minimizing non-linear least squares to identify ideal decisions. The final image is set by altering camera position and angle until a local minimum calculated by the cost function is met. Arev et al. applied such an automatic system to the real life concept of social cameras, which are carried by groups of people involved in the same activity [APS*14]. A trellis graph with all possible camera poses is constructed, forming an objective function that maximizes coverage of the important content in the scene. It is further weighted with constraints on cinematic guidelines and style parameters. Dynamic programming is then used to estimate the joint attention of all possible camera poses. Camera path planning is also widely explored in the context of unmanned aerial vehicles (UAVs). Smith et al. developed a framework for urban scene reconstruction by scanning the environment by UAVs [SMGH18]. To obtain the best possible images, a path is planned in two steps. First, scene data is gathered by an initial nadir grid pattern with 80/80 image overlap, that is transformed bit by bit into a complete nadir capture. The actual environmental reconstruction from the images is supported by Fuhrmann et al.ś *MVE platform* [FLG14]. Based on the result, the view and path planning step can be performed. Starting with a preset camera network, the system creates spherical histograms to

determine the orientation that maximizes the view and thereby optimizes the 3D model of the scene. After several iterations of this optimization, an objective function calculates the optimal positions for the UAV to take pictures [SMGH18].

2.2.2. Occlusion

To avoid the visual blocking of important actions when automatically cutting, Burg et al. implemented a GPU-based rendering technique to compute an anticipation map in toric space, which has been adapted specifically for cinematic camera control [BLC20]. This map predicts occlusions for a continuous set of cameras and, thus, serves as a basis for optimized camera support of user-specific visual arrangements. Besides this technique from Burg et al. there are several reactive systems that have been analyzed by Christie et al. Most of these systems include ray casts from the cameras to a particular object of interest, calculating any interference along the way. An even better performance can be achieved by considering intersections between rays and bounding volumes, instead. Real time computations can be achieved by projecting the bounding boxes of potentially occluding geometry onto a discretized sphere surrounding an object. The projections are then converted to global space and negated to represent an occlusion-free view on the object. Moreover, there are various target-tracking techniques as well as the idea to render the scene in hardware stencil buffers with colors associated to objects. An exceptional application to all methods analyzed by Christie et al. is for virtual endoscopy.

To prevent mistakes when the operating field is obscured, automatic camera systems can provide the best possible view [CON08].

2.2.3. Cutting

Christie et al. set up a framework for weighting shots and cuts for cartoons [CLR12]. The shot score is based on *Hitchcock*'s principles [Hit07], whereas the score for transitions originates from other common practices in film and television. The score is calculated as the sum of different ratings for specific actions, visibility and the composition of the characters. The action score is diminished by actions missed in a given fragment, visibility is defined by the overlapping area between projected bounding boxes and generally, shots are favored in which the actors are given more space relative to the whole image frame. In the transitions, continuity plays an important role. Screen continuity stands for the preference of transitions which maintain the actors' eyes at the same screen locations, gaze continuity penalizes camera flips, that cause apparent reversals in actors' gaze directions and lastly, motion continuity penalizes, if the actor's motions are reversed. Furthermore, the duration of shots is weighted to stick to the cinematic principles. The eventual selection of a view is done by traversing a graph constructed from all of this information with a best-first search for the lowest cost path [CLR12]. A real life application of a cut system has been explored by Kaiser et al. [KWK*12]. They created a technique for live event broadcasts called *FascinatE*, which serves as a virtual director getting input in the form of gestures that provoke zooming, panning, pausing or adjusting volume. *FascinatE* is able to analyze the surroundings and detect persons, salient regions, and audio events. A production scripting engine makes it possible to automatically select the best camera and the most relevant action at any time. This rule-based approach is co-acting with some constraints on cinematic principles, which are implemented in form of a *production grammar* which determines the cameras' placement, whether they are moving or static as well as their zoom, size and speed. It also describes when and how to cut. This system's greatest drawback is its restriction to a single point of view and a non-adjustable focus. Another automatic camera control approach was realized by Jhala and Young [JY05]. Their work tackled issues such as an occlusion-free view, the selection of ideal camera positions and angles, as well as the calculation of a view's clarity with respect to spatial and temporal coherence. However, their work focusses on supporting narrative and dialogues. To this end, a movie's story line has to be specified beforehand in form of an action sequence. Its combination with further annotations of story and characters is captured in an abstract data tree, which allows for path planning, which in turn informs a succession of camera directives, representing the cinematic schema. The calculation of an optimal path is based on matching local nodes and sub-trees with desirable features of the narrative. The system is supported by *Mimesis* [YRB*04], a service-oriented architecture for intelligent control of narratives in virtual environments [JY05].

As much as individual aspects of the approaches outlined in this section can support a music-driven cut system, they cannot be directly used for this purpose. In order to guide the cinematic performance, e.g. determining the cuts' parameters as well as their frequencies [CLR12], several of the presented techniques require extensive information about the narrative [PJSH13, JY05, ER07], occurring actions [DZ94], dialogues [HCS96], or game play events [Mee20]. These are not the decisive elements we want to consider, but rather we aim at a cinematic performance driven by music. The continuous nature of the latter mentioned game play events aligns best with our goals. Yet, the proposed approaches partially caused disorientation and breaks in presence [Mee20]. For our approach, the problem of occlusion shots as in [CON08, FLG14] did not play a significant role, either. As we aim at capturing sets of moving objects, temporarily and partially occluded views are not avoidable, but they mostly do not obfuscate the views, either. Likewise, intelligent path planning algorithms that rely on static environments, as for instance presented in [SMGH18], can also be omitted within our project. Overall, the preceding works mainly informed our approach with respect to (1) the need and concrete realisation to encode cinematic guidelines, and (2) the design of a decision making component to drive an artistic camera control.

3. FRAMEWORK DESIGN

The music-driven cut framework we propose is comprised of three parts (fig. 1). Its data basis is provided by (1) the audio analysis component. It extracts and makes available the frequency, amplitude and beat of a read input file or stream. (2) The cut system makes decisions based on the provided data. It decides which cutting technique, angle and camera movement are chosen and how the respective choices are parameterized. At the same time, a new location to cut to is chosen by (3) the view system. To put all of this into effect, the features of the UE4 engine, especially its *Cinematic Camera* component and the concept of *Level Sequences* were utilized together with the *fmod* api [Tec95] for conducting the audio analysis. The output of our system after completing its configura-

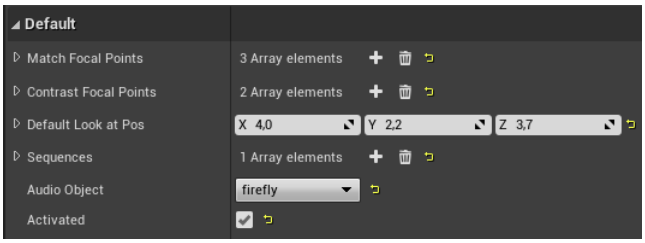

Figure 2: *Initialization example of a Focal Point.*

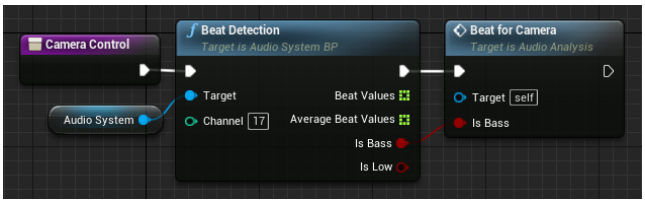

Figure 3: *Blueprint example for a camera call by the beat.*

tion can be experienced by simply pressing the play button in the UE4. Each time, a new film is created. The resulting movie then can either be recorded by a third party screen casting application or directly generated within the UE4 by writing to the master sequence of the scene. In this section, we will detail the three components of our proposed framework and their technical realization.

3.1. PREREQUISITES

Two basic design decisions need to be taken to provide a stage for the framework we propose. Aside from the choice of a song, the scene where the cinematic story unfolds, has to be arranged upfront. For this, different *Focal Points* have to be setup in the environment, one being labelled the default to which the view system can fall back if not other information is available. They can be considered virtual cameras linked to and focusing on specific objects. In the UE4 engine, *Focal Points* can, for instance, be setup as *Cinematic Cameras* linked to one of a set of available "audio"-enabled objects, which we refer to as *audio objects*. Dependent on which of the *audio objects* a *Focal Point* is linked to, it requires a definition of its relationship to other *Focal Points*. This includes the "Contrast" property, which is a list of all *Focal Points* with *audio objects* that look or behave contrary to the *audio object* assigned to the specific *Focal Point*. Likewise, there is a "Match" property, with a list of *Focal Points* with similar *audio objects*. *Focal Points* may further be expanded by a set of different dolly shots. In UE4, these can be realized by means of so-called *Level Sequences*, which provide a rich toolbox to animate arbitrary data attributes. The provided degrees of freedom for setting the stage allow for a wide variety of individual cinematic looks and story lines. In order to avoid cutting to a humdrum view or inanimate objects, only those *Focal Points* that target objects undergoing significant changes are considered as cutting targets. To this end, activity thresholds can be associated with individual animated properties.

Focal Points can also be marked as 'highly active', if the objects' changes are particularly interesting, e.g. when an object has started moving for the first time. Such highly active *Focal Points* are prioritized in the selection process.

3.2. AUDIO ANALYSIS

Throughout the entire movie making process, our framework analyzes the music stream to extract the amplitude, the frequency and the beat at every frame. It can distinguish different channels, which makes it possible to individually keep track of specific instruments or themes. Therefore, in order to make sure the system is properly prepared, the developer needs to add one music file for each channel (the raw .wav format is currently supported). Different files/channels may also be joined and fed into a single, composite channel. In order to detect the musical features, a virtual *fmod* project is created when starting the playback. Within a so-called *fmod bank*, a dedicated channel is setup for each provided file. For the analysis, digital signal processors (DSP) are utilized, which allow for raw pulse code modulation samples to be processed in order to alter the sound, or in the given case, obtain information. A DSP is assigned to each channel. It considers a specified bucket size to sample the audio spectrum in several frequency domains. Its collected data can be polled on-demand. The underlying concepts of this audio analysis are introduced in [BL10]. We adopted the implementations from [Cla16] and [BF18], made them work together and adapted them to our needs.

To simplify the setup and work flow for designers, we made the functionality of the audio analysis available in UE4's blueprints, i.e. its visual programming environment. There, we provide a high-level access to the music definition, beat detection, amplitude and frequency analysis. The only necessary input for a retrieving analytical information is the channel number and the bucket size. The received values can then be used to alter and modify the objects in the scene or just to trigger a cut. In this context it is imperative to prohibit too fast cutting sequences solely dependant on the music. Cutting every second, for instance, would be quite irritating to the viewer. Our system includes a default minimum waiting period of 3 seconds, that addresses this problem. Yet, the developer is allowed to adjust this waiting period as he or she requires. To add some further variability, the designer can specify a set of maximal view durations that will be randomly applied at runtime.

3.3. CUT SYSTEM

The cut system starts as soon as supplied with the analytical data. It considers three aspects to determine the necessary cutting parameters: (1) The previous cut choice, (2) cinematic principles taken from the seminal book on film language by Arijon [Ari91], and (3) a certain amount of randomness. First, one cutting type from the following ones needs to be determined.

- A *Direct Cut* is a standard cut without any special characteristics.
- A *Match Cut* implies that the next view matches the current view to some extent.
- A *Contrast Cut* is the opposite of the *Match Cut*, which means the new view is contrasting the current one to some extent.
- A *L-Cut* is a standard cut, but audio-delayed. The audio of the

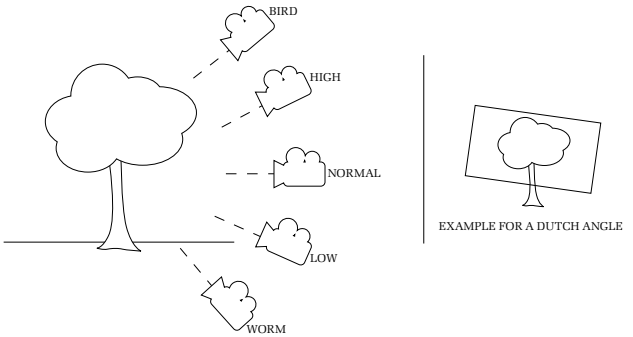

Figure 4: *Different angles of a camera on an object in the scene.*

Figure 5: *Screenshot from a demo movie: The tree is moved by the wind, which is controlled by a synthesizer. The morphing cloud reacts to a snare drum.*

current scene is still played while the view already cuts to the next scene.

- A *J-Cut* works the other way around than the *L-Cut*. I.e. the sound from the next scene is already audible, but the view is set with a temporal delay.
- A *Cut In* makes the camera stay in place, but it zooms in very strongly.
- A *Cut Out* opposes *Cut In* by strongly zooming out.

Two subsequent cuts may not be of the same type. Additionally, the successors for *Cut Ins* and *Cut Outs* need to be the respective opposites, implementing the rules outlined by [Ari91], and the third cut in such a sequences is restricted to the other choices above. After the choice of the cutting type, the camera angle is chosen from the following options (depicted in figure 4): *Normal view*, *high-angle shot*, *low-angle shot*, *bird's-eye view*, *worm's-eye view*, and *dutch angle*. As before, two subsequent angles may not be the same. In a third step, the cutting system determines whether a dolly shot should be used to animate the transition to the new cut, or not.

3.4. VIEW SYSTEM

The view system is responsible for (1) selecting the next *Focal Point* and (2) realising the cut calculated by the cut system. In case the cut system decided for a *Cut In* or *Cut Out*, the new *Focal Point* is set to be the current one, as the location of the camera stays unchanged. and only zooms in or out. If the cut system has chosen a *Contrast Cut* or *Match Cut*, the choice for the new *Focal Point* is made randomly between the respective candidates from the configuration of the current *Focal Point*. For all other cut options, the next *Focal Point* is selected from those whose associated audio objects is currently active, whereas highly active audio objects result in a higher probability of selection. Thus, it is randomly selected between all highly active objects, in case there are none, the choice falls on one of the active ones, also at random. After the determination of the next *Focal Point*, its camera position and orientation is adjusted to fit the previously selected camera angle. To obtain a sharp image, the focus setting of the camera is adjusted, depending on its new location and its looked-at *audio object*. If the cut system has chosen a static camera, the process is finished. But if the choice was to put the camera in motion, a last random selection has to be made among all predefined dolly shots of the specific *Focal Point*.

4. DEMO INSTANCE

As we only started experimenting with its general idea, we tested the framework with a very simple outdoor scene composed of objects that would usually be static. By virtue of the rhythmic input extracted by the audio analysis, they come to life alongside some environmental elements. Every object represents a musical instrument, simulating a band performance. Whenever the specific instrument is particularly prominent in the song, the movie cuts to the respective *Focal Point*. A tree is standing in the middle of a hilly landscape, whose branches are set in motion by wind and small impulses, causing leaves to fall down (fig. 5). The intensity of the wind is modified by the frequency values of a synthesizer, whose amplitude values in turn control two rags floating through the scenery (fig. 6). The impulses for the tree are caused by the amplitude values of an electrical guitar, also reshaping the tree's branches by scaling them up and down. Additionally, the frequency values of a second guitar are responsible for the movements of a swarm of fireflies buzzing around a small lake. A drum kit is split into the individual elements, which influence the scenery in different ways. The beat of the bass drum is connected to some glossy spheres bouncing and popping around on the lake, a cloud is listening to the beat of a snare drum together with the frequency values of the synthesizer and a ladder is able to alter its height contingent on the beat of a drum kit's hi-hat (fig. 7). Lastly, a mirror is being tilted by the beat of a musical kick (fig. 8). This kick is also in charge of triggering the cutting itself. Hereby, the scenes switch between close and far views of the moving objects as well as a static camera and dolly shots, due to the random choices between the valid options. The song starts with few instruments, but more begin to play after a few seconds. This translates to the little movement in the beginning with less cuts, as there are not many active *Focal Points* yet. Later, this changes to quite an energetic cutting switching between the different the techniques, as the intensity of the music increases more and more over time.

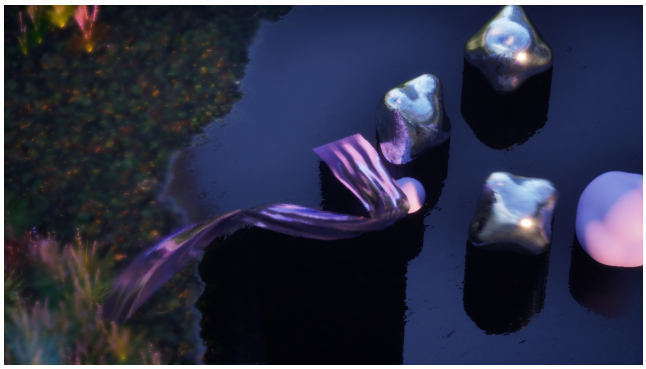

Figure 6: *Screenshot from the demo movie: A rag floats through the scene, driven by the wind. The glossy spheres bounce in synchronization with a bass drum.*

Figure 7: *Screenshot from the demo movie: The ladder's height is changed relative to a drum kit's hi-hat.*

An example rendering with a cut option we selected can be watched here: https://vimeo.com/508517056.

5. DISCUSSION

The sample project demonstrates that the overall concept can lead to an aesthetic result. Still, a closer look at its generated details in combination with the underlying processes is needed to better understand the pros and cons of our proposed approach, and the capabilities and limitations of the implemented framework. Currently, it is required to include multiple cameras within the scene and to manually set them up. It might be easier to only add one camera, which can be displaced and manipulated throughout the recording. But the somewhat labour-intensive initialization pays off: It allows for highly individualised design of a story line well-aligned with boundaries of cinematic parameters. It also results in a great replay value as the generated movie varies at each run - respecting both general cutting rules and the individual style introduced by the designer. Yet, at this point, further explorations are required to understand which design details might be automatised without loosing desirable control or diminishing the replay value. Currently, the audio analysis is limited to amplitude, frequency and beat detection. With an increasing number of objects being animated by the music, the dynamics of the movie would eventually be lost since then all objects depend on the same parameters. This problem could be solved by extracting more parameters from the audio stream such as pitch, by introducing different random number generators (RNGs) and storing different RNG seeds for different channels or decision processes. Timing of the cutting process might come into conflict with dolly shots or animations of audio objects. Especially dolly shot interruptions are hard to cope with since they are currently defined before playtime. For instance, an early cut in the middle of a focus shift would be rather unpleasant. This issue could be tackled by implementing a look-ahead for the audio analysis, so that the system knows when the next cut will occur and select the current cut accordingly. Furthermore, the currently deployed, great amount of randomness may lead to somewhat sub-optimal locations of the camera. Despite cutting to a highly active *Focal Point*, the adjustment of the camera to fit a chosen angle might result in an odd-looking view and the focused audio object might be presented in a strange perspective. Although this could be solved by adding more constraints on the view on some of the objects, this in turn would interfere with the movie's flow. As the way an *audio object* cannot be foreseen, only a very far view would always frame the action, limiting the result to fewer camera angles and cut techniques. Testing and refining the framework, the distinction of low and high activity of audio objects proofed rather practical. It allows for dynamic prioritization of views and as the music is in charge of the objects' motions, the system produces very unique aesthetics. Considering this observation, a continuous prioritisation might yield further improvements.

6. CONCLUSION AND OUTLOOK

In this paper, we presented (1) an approach to automatically translate music streams to an animation story line supported by cinematic cutting. To implement this approach, we created (2) an accessible framework whose code base integrates music analysis by means of *fmod* and wraps low-level functionality into high-level, visually programmable components in the Unreal Engine's blueprint scripting environment. Finally, we demonstrated how the approach could drive a (3) specific artistic, music-driven movie instance. With the help of with the *fmod api* we analyze a music stream in real time and utilize extracted music features parameters to control the behaviour of the objects in the scene as well as trigger the cutting between views. For this, a specific cutting technique, angle and movement of the camera is chosen with regard to common cinematic guidelines. The new view into the scene is further dependant on the activity level of the objects.

What could boost the current system to a higher level is an extended audio system with additional descriptive parameters and live music. The *fmod* software is capable of processing live input, but the sound is completely merged into one group channel in UE4, which would have limited the audio analysis tremendously. Further investigations and tests on the interplay between *fmod* and UE4 as well as other audio analysis platforms to solve this issue would be beneficial. Another important aspect we plan to implement is the look-ahead to increase predictability, which would, in turn, lead to a clearer separation between activity levels and a better, more

Figure 8: *Screenshot from the demo movie: The mirror is tilted with a musical kick.*

thoughtful view choices. We intend to address all of this by creating an engine plugin, which will mitigate the work load of the setup and also simplify the process of manipulating objects by means of audio information. The movie designer would not have to write code, but could only set and weigh the respective correlations in the editor. Then, it would also be easier to test the system with movies of different styles and scenic setups, to get an even better insight on its artistic variations of the result. Overall, our approach has the potential to extend the power of the editing process: Cutting may be performed completely automatically and in synchronization with the beat. Yet, individually granted degrees of stochasticity or definition of probabilistic rule-sets to implement different paths of best practice cutting strategies can render the experience of one single music video definition repeatedly surprising and attractive. It also allows the designer to (automatically) provide customizations for his or her different audiences.

References

[APS*14] AREV I., PARK H. S., SHEIKH Y., HODGINS J., SHAMIR A.: Automatic editing of footage from multiple social cameras. *ACM Transactions on Graphics (TOG) 33*, 4 (2014), 1–11. 3

[Ari91] ARIJON D.: *Grammar of the film language*. Silman-James Press, 1991. 5, 6

[BBB*06] BARDZELL J., BARDZELL S., BRIGGS C., MAKICE K., RYAN W., WELDON M.: Machinima prototyping: an approach to evaluation. In *Proceedings of the 4th Nordic conference on Human-computer interaction: changing roles* (2006), pp. 433–436. 2

[BF18] BARRIO C., FUENTES C.: Beat detection algorithm, 2018. URL: https://www.parallelcube.com/2018/03/30/beat-detection-algorithm/. 5

[Bio02] BIOWARE: Neverwinter nights, 2002. 2

[BL10] BOULANGER R., LAZZARINI V.: *The audio programming book*. the MIT Press, 2010. 5

[BLC20] BURG L., LINO C., CHRISTIE M.: Real-time anticipation of occlusions for automated camera control in toric space. In *Computer Graphics Forum* (2020), vol. 39, Wiley Online Library, pp. 523–533. 4

[Cla16] CLABORN C.: Fmod Audio Engine, 2016. URL: https://codyclaborn.me/tutorials/making-a-basic-fmod-audio-engine-in-c/. 5

[CLR12] CHRISTIE M., LINO C., RONFARD R.: Film editing for third person games and machinima. In *Workshop on Intelligent Cinematography and Editing* (2012), ACM. 4

[CON08] CHRISTIE M., OLIVIER P., NORMAND J.-M.: Camera control in computer graphics. In *Computer Graphics Forum* (2008), vol. 27, Wiley Online Library, pp. 2197–2218. 4

[DZ94] DRUCKER S. M., ZELTZER D.: Intelligent camera control in a virtual environment. In *Graphics Interface* (1994), Citeseer, pp. 190–190. 3, 4

[Ent04] ENTERTAINMENT B.: World of warcraft, 2004. 2

[ER07] ELSON D. K., RIEDL M.: A lightweight intelligent virtual cinematography system for machinima production. 2, 4

[FLG14] FUHRMANN S., LANGGUTH F., GOESELE M.: Mve-a multi-view reconstruction environment. In *GCH* (2014), Citeseer, pp. 11–18. 3, 4

[Gam98] GAMES E.: Unreal engine, 1998. 1

[HCS96] HE L.-W., COHEN M. F., SALESIN D. H.: The virtual cinematographer: A paradigm for automatic real-time camera control and directing. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques* (1996), pp. 217–224. 4

[Hit07] HITCHCOCK C.: Prevention, preemption, and the principle of sufficient reason. *The Philosophical Review 116*, 4 (2007), 495–532. 4

[HZ12] HU W., ZHANG X.: A semiautomatic control technique for machinima virtual camera. In *2012 International Conference on Computer Science and Electronics Engineering* (2012), vol. 1, IEEE, pp. 112–115. 2

[Int96] INTERACTIVE G.: Quake, 1996. 2

[JY05] JHALA A., YOUNG R. M.: A discourse planning approach to cinematic camera control for narratives in virtual environments. In *AAAI* (2005), vol. 5, pp. 307–312. 4

[KWK*12] KAISER R., WEISS W., KIENAST G., BAILER W., THALER M., THALLINGER G.: *Automatic Camera Selection for Format Agnostic Live Event Broadcast Production*. na, 2012. 4

[Lab03] LAB L.: Second life, 2003. 2

[Low08] LOWOOD H.: High-performance play: The making of machinima. *International Journal of Technology Management & Sustainable Development 7*, 1 (2008), 25–42. 2

[Mas65] MASCELLI J. V.: *The five C's of cinematography*, vol. 1. Grafic Publications, 1965. 3

[Max04] MAXIS E. G.: The sims 2, 2004. 2

[Mee20] MEEDER I. H.: Intelligent cinematic camera control for real-time graphics applications. 3, 4

[MY09] MUNILLA S., YOUNG R. M.: Zuzen, a cloud-based framework for automated machinima generation. In *Proceedings of the International Conference on Advances in Computer Enterntainment Technology* (2009), pp. 199–206. 2

[Pic06] PICARD M.: Machinima: Video game as an art form. CGSA 2006 Symposium. 2

[PJSH13] PRIMA D. A., JAVA B. B. F., SURYAPTO E., HARIADI M.: Secondary camera placement in machinema using behavior trees. In *2013 International Conference on QiR* (2013), IEEE, pp. 94–98. 2, 4

[SMGH18] SMITH N., MOEHRLE N., GOESELE M., HEIDRICH W.: Aerial path planning for urban scene reconstruction: a continuous optimization method and benchmark. 3, 4

[Stu04] STUDIOS B.: Halo 2, 2004. 2

[Stu05] STUDIOS L.: The movies, 2005. 2

[Tec95] TECHNOLOGIES F.: Fmod, 1995. 4

[Tho00] THOMPSON A.: Virtual stage, 2000. 2

[UB07] UDAR S., BORGAONKAR R.: Darshak, 2007. URL: https://github.com/darshakframework/darshak/. 2

[YRB*04] YOUNG R. M., RIEDL M. O., BRANLY M., JHALA A., MARTIN R., SARETTO C.: An architecture for integrating plan-based behavior generation with interactive game environments. *J. Game Dev. 1*, 1 (2004), 1–29. 4