A Novel Approach to 3D Storyboarding

Federico Manuri[1(✉)], Andrea Sanna[1], Marco Scarzello[1], and Francesco De Pace[2]

[1] DAUIN, Politecnico di Torino, corso Duca degli Abruzzi 24, 10129 Torino, Italy
federico.manuri@polito.it
[2] Institute of Visual Computing and Human-Centered Technology TU Wien, Vienna, Austria
http://grains.polito.it

Abstract. Creatives in the animation and film industries constantly explore new and innovative tools and methods to enhance their creative process, especially in pre-production. As realistic, real-time rendering techniques have emerged in recent years, 3D game engines, modeling, and animation tools have been exploited to support storyboarding and movie prototyping. This research proposes a 3D storyboarding tool to improve existing storytelling approaches. A novel storyboarding pipeline is proposed, which can automatically generate a storyboard including camera details and a textual description of the events occurring in each scene. Users create storyboards by selecting actors, performing available actions, positioning the camera in the 3D scene, and taking screenshots to save a vignette of the storyboard; the corresponding description is generated based on the actors' actions and their status. A software implementation of the proposed pipeline has also been developed in the guise of a 3D desktop application aimed at expert and novice storyboarders. The system has been tested to evaluate its usability. Preliminary results confirm that the users have appreciated the application.

Keywords: 3D storyboarding · character animation · authoring tool

1 Introduction

Creatives in the animation and film industries constantly explore new and innovative tools and methods to enhance their creative process, especially in pre-production. Traditional approaches rely on hand-drawn storyboards and physical mockups, whereas information technology introduced sketch-based [1] and picture-based 2D [2,3] drawing applications. The production phase of cinematic and computer-generated imagery (CGI) sequences usually includes state-of-the-art technologies. However, the pre-production phase usually relies on more traditional methods. After completing the script, storyboard artists realize 2D panels for each shot, drawing the background, the foreground, and the characters. However, as 3D animation became popular, 3D artists were employed to recreate in

M. Clayton et al. (Eds.): INTETAIN 2023, LNICST 560, pp. 178–192, 2024.
https://doi.org/10.1007/978-3-031-55722-4_13

3D the storyboard drawings. This approach is also helpful in cinematic production to pre-visualize the story before the video shoot phase. Some commercial applications for 3D pre-visualization exist, such as Frameforge 3D [4] or ShotPro [5], but these solutions are usually complex to master and present a steep learning curve. Hand-drawn storyboarding has the advantages of an interactive process, as the drawings can be corrected immediately as the lines appear on the paper (or screen). However, the conversion accuracy from 2D images to a 3D scene can only be judged when the 3D artists complete their works: actual production statistics confirm that up to half of the layout shots may need a retake [6], as framing discrepancies occur. Even the best storyboard artist draws only an approximation of a camera shot: it would be challenging to draw a scene as it would look through a specific camera lens with 100% accuracy. Drawing the correct scale of objects without absolute references (for cinematic production) or 3D references (for CGI productions) is another possible mistake. Performing the storyboarding process in 3D can resolve these issues and provide an interactive pre-visualization of the storyboard [7].

This research proposes a 3D storyboarding tool to improve existing storytelling approaches. A novel storyboarding pipeline is proposed, which can automatically generate a storyboard including camera details and a textual description of the actions performed in three-dimensional environments. A software implementation of the proposed pipeline has also been developed in the guise of a 3D desktop application aimed at expert and novice storyboarders. The application enables the user to create a 3D virtual stage starting from an open 3D world: users can add environment tiles to define different locations and populate them with structures, objects, and actors (e.g., humans or animals). Each actor is seen as a state machine: a list of available actions is provided based on the current actor's state and its position in the scene. Once the 3D virtual stage is complete, users can start creating the storyboard, selecting actors, performing actions based on their state, and taking screenshots to save a vignette of the storyboard and the corresponding description. The description is automatically generated based on the actors' actions and their status.

This paper is organized as follows: Sect. 2 provides an overview of the state-of-the-art, whereas Sect. 3 depicts the system design. Section 4 describes in detail the software implementation. Section 5 presents the test performed to evaluate the proposed system and the analysis of the results.

2 Previous Works

As realistic, real-time rendering techniques have emerged in recent years, 3D modeling and animation tools and 3D game engines have been researched and explored to support storyboarding and movie prototyping. A first attempt to automatically generate storyboards was developed by Pizzi et al. [8]: then authors proposed an authoring tool allowing game designers to formalize, visualize, modify, and validate game-level solutions in the form of automatically generated 2D storyboards. The proposed system features planning techniques to

plan level solutions, with the main agent corresponding to the player character, game actions to planning operators, and level objectives as goals. The *Director's Lens* instead is a first attempt at exploiting 3D to innovate the shooting process of CGI through assisting filmmakers in camera composition [9]: the proposed system employs an intelligent and automatic cinematography engine that can compute a set of suitable camera placements for starting a shot. The proposed method can potentially simplify the process of shooting CGI, resulting in a novel workflow based on the interactive collaboration of human creativity with automated intelligence. In [7], the authors investigate a novel way to automatically pose 3D models from 2D sketches: the pose inference is formulated as an optimization problem, and a parallel variation of the Particle Swarm Optimisation algorithm has been proposed to search for the minimum of its objective function. The scope of this research was to pose models for pre-visualization, providing a direct link between the storyboarding and the pose layout phases of a 3D animation pipeline. A computer-assisted narrative animation synthesis (CANVAS) authoring tool was proposed by Kapadia et al. in [10] for synthesizing multi-character animations from sparsely specified narrative events. Given the key plot point in a story, the proposed system automatically resolves incomplete story definitions to produce a consistent and complete narrative, filling in the missing details necessary to synthesize a 3D animation that complies with the author's constraints. The proposed system provides an accessible interface for rapidly authoring complex narratives, suitable for pre-visualization or cutscenes. In [11], the authors propose a novel approach for the automated staging of actors and cameras in a virtual 3D environment given a set of constraints specified with a dedicated staging language that extends the Prose Storyboard Language (PSL). The system resolves complex spatial relations such as visibility or character framing, considering the relations that link cameras and entities in the scene to position the characters and the cameras simultaneously. PSL uses a simple syntax and limited vocabulary borrowed from working practices in traditional movie-making, intended to be readable by machines and humans, and has been designed over the last ten years to serve as a high-level user interface for intelligent cinematography and editing systems [12]. Kim et al. [13] developed a tool for Auto-generating Storyboard And Previz (ASAP) for screenwriters and filmmakers: the proposed system enables users to easily simulate stories as 3D animated scenes with virtual characters in a 3D environment. Providing a script that follows the Final Draft screenwriting tool format, the system parses it, identifying actions, characters, and dialogue, using a combination of deep learning, data-driven, and rules-based approaches. The system output consists of automatically generated pre-visualized animations, simulating natural behaviors and realistic dialog scenes. Furthermore, users can create storyboard shots by capturing scenes being played. In [14], the authors propose an open-source, semi-automated cinematography toolset capable of procedurally generating in-game cutscenes in the style of a specific movie director. The system has been developed with the Unity game engine [15]. It combines run-time cinematography automation with a novel timeline and storyboard interface for design-time manipulation, allowing

for cutscenes to unfold dynamically based on the game state. Chen et al. [16] proposed a sentiment-aware generative model for visual storytelling based on the sequential vision-to-language approach, which generates coherent and rich image descriptions from a sequence of input images through a multi-layered sentiment extraction module based on deep learning models. Overall, research efforts in recent years are primarily aimed toward automatic generations of storyboards, either in 2D or 3D, starting from the script or other text-based constraints. Some works focus on specific tasks, such as positioning the camera in the 3D scene or providing different camera shot positions. Recent works further exploit artificial intelligence to automatically generate 3D animations or poses from the script or the 2D storyboard as a pre-visualization technique. Others focus on automatically developing the most coherent description of an image through deep learning models. However, artists are still limited to a few options for creating storyboards, including sketch-based or picture-based approaches.

The scope of the proposed system is to investigate a novel approach to storyboarding. The classic process consists of drawing each storyboard shot in 2D: this could be done with either sketch-based or picture-based tools. Users with good drawing skills mostly appreciate sketch-based tools due to the freedom of expression provided by this approach. Instead, those not skilled in drawing generally prefer picture-based tools since they use readily available pictures as building blocks, resulting in less freedom but faster content creation than the previous approach. 3D storyboarding applications usually provide a different paradigm, starting from a set of rigged characters instead of static 2D images and allowing users to pose the characters through their armatures. However, posing a character properly could be difficult and time-consuming, especially for users who are not skilled in 3D animation. The proposed approach lets the user move the character in the scene freely using a video game-based approach, using a combination of keyboard and mouse, likewise first-person and third-person shooter video games. However, instead of posing the character to define their appearance, the user can choose which action the character should perform from a list. The list of available actions depends on the context, the proximity to other objects/characters, and the state of the actors involved; an animation is provided and automatically played upon selection. Actions performed by the user are saved in chronological order as events and are used to generate the descriptions for each vignette of the storyboard automatically. Moreover, the user can move the camera to decide the framing and change the camera's focal length of the lens, which is then reported in the storyboard. Traditional methods require the user to find the best way to convey the script's text in a meaningful, graphical representation enhanced with camera details and additional textual information. The proposed approach lets the user enact the story in a 3D scene, similar to a sandbox video game, shifting the user's attention from the final result to obtain (the storyboard) to the story to play (the script).

The aim of the proposed system is to enable the user to play out the story in a 3D environment, controlling the available characters, trying out different camera views, and automatically obtaining both the camera information and

the textual description of the actions happening in each scene. This should help the user focus on the story and define the actions' timing. Thus, the first step should be to enable the user to select the characters available in the scene, change their position, move the camera, and generate a vignette. This can be easily obtained in a 3D game engine such as Unity 3D [15]; however, if the only action available to the characters is to move, obtaining a detailed description of the scene in the storyboard would not be possible. Moreover, there would be no novelty compared to traditional techniques, since the user should focus on the graphical output of the actions without the characters performing them.

3 Proposed System Design

This section describes the design phase of the proposed storyboard pipeline, based on the insight obtained from the analysis of the state-of-the-art.

3.1 3D Virtual Stage

Even if the focus of the research project is to provide a novel approach to storyboarding, the setup of the 3D virtual stage represents a necessary condition. This step should exploit straightforward techniques to create a 3D scene creating locations and adding objects and characters. Based on the state-of-the-art, the system should enable the user to:

- select different kinds of tiles, e.g., stone, wood, and soil, to texture the floors;
- add objects to the scene, either to:
  - define locations, such as rooms, courtyards, streets, etc., through walls, doors, stairs, etc.;
  - furnish existing locations;
- add characters to the scene, defining their starting position.

Once all the elements described in the script are available as 3D assets, the user can start storyboarding.

3.2 Actors' Actions and Animations

The following step is to determine how the user can perform different actions with the actors. This offers two possible options: unconstrained actions versus constrained actions. The first paradigm lets the user perform any possible actions, either selecting them from a list or typing them in a text box; no constraint is applied, neither in terms of environment, objects or characters proximity, or acting character conditions. The most critical problem of this approach regards the consistency of the story since the users may perform actions that make no sense in the given context: performing an action that involves a character or object not available (e.g., closing a door without a door in the scene); performing an action which is inconsistent with the actor status (e.g., swimming while driving a car, going to bed while eating in the kitchen). A secondary problem of an

unconstrained approach regards the user interface: to select the proper action, the user should either navigate a long list or complex menu to find the right one or write it down.

On the contrary, a constrained approach means that only the actions available to the characters could be performed based on their status, position in the scene, and proximity to other objects or actors. All the elements in the 3D scene are divided into two groups: invariable elements, whose status is immutable, and variable elements, which may change depending on the actor's actions. Variable elements are either active or passive: active elements (usually characters) can be selected by the user to perform actions, whereas passive elements can only undergo status changes due to active elements' actions. This way, not only the user interface would provide a limited list of available actions, but the actions affecting others, either objects, actors, or the environment, would result in changes in the scene. For example, moving a pot from point A to point B would require the character to move near point A, pick it up, move to point B, and put down the pot. Moreover, since one of the aims of the storyboard is to obtain a graphical representation of the events of a given scene, the user should be able not only to select an action from a list depending on the current character status but also to see a corresponding animation providing a graphical representation of such action. Such transformations to the 3D scene are essential to obtain an accurate visual representation, focusing on the story (what should happen) instead of the storyboard (what should be visible in the storyboard).

3.3 State-Machine Representation

The simplest way to develop a constrained approach involves adopting a finite state machine (FSM) strategy. An FSM is a mathematical model which describes a system with a finite number of states as a graph. Each graph node represents a status, and the arrows connecting the nodes determine the transition between one status and another in response to some inputs. An FSM is denoted by its initial state, the list of its states, and the inputs that trigger each transition. To apply an FSM approach to the proposed scenario, all the elements in the scene should be appropriately classified as invariable, active (variable), or passive (variable). The behavior of each variable element is represented with an FSM (characters and objects). FSMs are frequently used in video games to define playable and non-playable characters. For example, a player may be in the wandering status as long as the enemies are far away and automatically enter the combat status (e.g., drawing weapons) when the enemies are nearby and vice-versa. The mathematical model of a deterministic finite-state machine is a quintuple $(\Sigma, S, s_0, \delta, F)$ where:

- Σ is a finite, non-empty set of symbols, representing the input alphabet;
- S is a finite, non-empty set of state;
- s_0 is the initial state, with $s_0 \in S$;
- δ is the state-transition function $\delta : S \times \Sigma \rightarrow S$;
- F is a subset of S, possibly empty, of final states.

FSMs can be described using State Chart XML (SCXML), an XML-based markup language designed for state machine notation. SCXML can simplify the transition between the code used to handle the characters at runtime and a user interface that enables the user to define, change and update the FSM for a character. Figure 1 shows a graphical representation of a simple FSM for a human character.

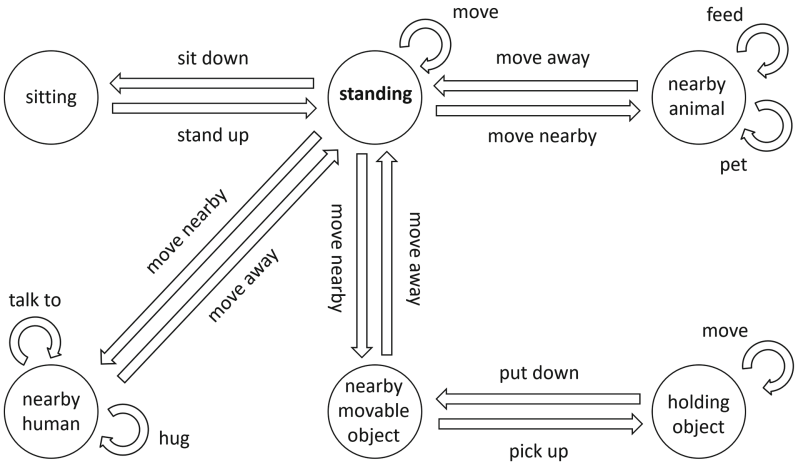

Fig. 1. A graphical representation of a simple FSM for a human character.

Given a 3D asset pertaining to the variable element category, it would be useful to infer its FSM transitions from the available animations list. Moreover, animations could be renamed to incorporate all the information necessary to create the corresponding FSM: every animation can be renamed as a sequence $N;S;E;T$ with N the action's name, S the list of compatible starting status, comma separated, E the list of consistent ending status, comma separated (or $SAME$ if the ending status should be equal to the starting one), and T the target of the action (either self or another variable element, active or passive). For example, $STARTRUNNING;IDLE,WALKING;RUNNING;SELF$, representing the character that may start running from on the three compatible statuses, or $GREET;IDLE,SITTING,WALKING;SAME;CHARACTER$, representing the character that may greet another character, starting from one of the compatible statuses and ending in the same status after the action. Passive elements, such as doors, may have a starting status of *close*, and two animations, *open* and *close*, that should be renamed as $OPEN;CLOSE;OPEN;SELF$ and $CLOSE;OPEN;CLOSE;SELF$, representing the fact that a door can only be open if its starting status is close, and it ends up open, and vice-versa. The element type, passive, should imply that an actor should perform the action. Using this approach, a text parser can be scripted to automatically define the FSM

of a variable element (both active and passive) based on the list of available animations.

3.4 3D Storyboarding

Constraining the user choices based on the environment and FSMs of the variable elements available in the scene should simplify transposing the story from the script to the 3D setting. The process of storyboarding in a virtual location would require the user to:

1. select one of the variable active elements (AE);
2. based on the script, choose between:
   - move the AE in the scene;
   - perform one of the available actions;
3. (optional) take a snapshot of the current camera view to generate a vignette of the storyboard;
4. restart from step 1.

Based on the FSM approach described in the previous section, generating a stack of events would be possible, adding one element to the stack each time the user acts. The $SUBJECT$, $ACTION$, $TARGET$, and $FRAME$ values should be saved into the stack for each event. The $SUBJECT$ value corresponds to the currently active actor; the $ACTION$ value is the one chosen by the user, whereas the $TARGET$ is either the active actor or another variable element in the scene; the $FRAME$ value should enable the user to add to the stack multiple events at the same time to describe concurrent events. Thus, the system should provide an interface to edit this value.

The user should be able to undo actions. However, since the system's consistency is essential, the editing of the stack should be handled with extreme care to avoid inconsistent states for one or more elements in the scene. Thus, the system should provide an undo action for the last event in the stack, rolling back to their previous status each element involved by the deleted event. On the other hand, randomly canceling events in the stack should not be possible: a very complex consistency check should be developed to verify the consistency of all the following events unless the system automatically deletes all subsequent actions.

3.5 Camera

In the proposed scenario, the camera would work as a variable active element that can only be moved in the scene. Usually, the storyboard's vignette may contain details on the camera focal lengths or the shot size. The system may provide a menu with camera options (instead of actions), such as a list of camera lens presets or a mechanism to adjust the focal length or the focus. All these camera properties can be easily conveyed in additional info added to the vignette description. On the other hand, the shot size, which depends on the distance

between the camera and the subject, would require additional computations to provide an accurate description or labeling in the final storyboard. If necessary, camera movements and variable parameters could be defined as an FSM as well; this could be useful if the user wants to track down in the vignette description not only the events occurring in the 3D scene but also the camera actions (e.g., movements or change of shot size).

3.6 Automatic Vignette Labeling

Once the user wants to save the chain of events in a single vignette of the storyboard, the system would generate a save-point in the stack; then, all the events in the stack following the last save-point should be converted into sentences to automatically label the vignette. A translation script generates a sentence from an event parsing the $SUBJECT$, $ACTION$, $TARGET$, and $FRAME$ variables saved for that element in the stack. The system also takes into account the timing information provided by the timeline to define if multiple actions pertaining to the same vignette occur simultaneously or consequentially and introduce temporal adverbs in the sentence accordingly.

4 Software Implementation

The design requirements described in the previous section guided the development of a prototype that implements the proposed 3D storyboarding methodology. Unity 3D has been chosen to deploy the system since it is a 3D game engine freely available for research. It is highly compatible with various visualization devices like desktop computers, mobile devices, and augmented and virtual reality headsets. WordNet [18] has been used to provide variability in the textual descriptions of the storyboard's vignettes. More specifically, the Syn.WordNet [19] package for Unity has been installed to request sets of synonyms (*synset*) to WordNet from the Unity scripts.

Figure 2 shows the pipeline for the proposed system. When the application starts, it loads both 3D assets and (if available) their FSMs and animations into the system. Then, the starting menu provides three choices to the user to (1) create or edit an existing 3D scene, (2) start storyboarding from an existing 3D scene, or (3) edit or create FSMs for the available 3D assets.

4.1 User Interface

3D Virtual Stage. The system enables users to add, move, or delete 3D assets to the scene. Figure 3 shows the user interface of the virtual staging (on the left). Three dropdown menus let the user choose among environment assets, objects, and characters. Selecting a 3D asset, the user can move, remove or rename it. Whereas move and remove can be obtained through shortcuts, rename is useful for changing characters' names and using them instead of the default value in the automatic text generation for the vignettes' descriptions. Once the 3D scene is ready, the user can save it, move back to the initial menu, or proceed with the storyboarding step.

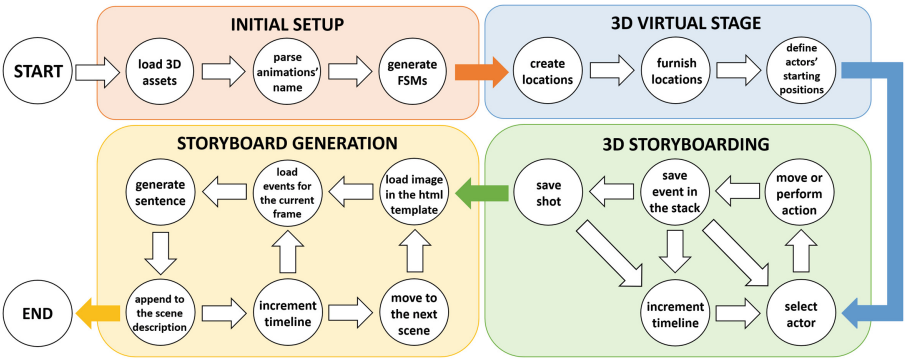

Fig. 2. The pipeline of the proposed 3D storyboarding tool.

Fig. 3. The 3D virtual stage interface (on the left) and the 3D storyboarding interface (on the right).

3D Storyboarding. In the storyboarding phase, the user enacts the story described in the script to automatically generate the corresponding storyboard. Figure 3 shows the corresponding UI for this phase (on the right).

When the user selects a character, the active character label in the bottom right corner of the screen is updated. The user can move the character in the scene using the keyboard through the WASD keys since it is a standard in 3D first and 3rd person videogames. The move action is saved in the event stack, considering the environment the user is moving on, defined by the floor's tiles, and the target the user is moving towards. The target is computed considering the character's forward direction when the movement ends and tracing an orthogonal line that stops at the first 3D assets available in the scene. Then, the user can again select the active character to show the available actions menu. When the character moves inside the range of an interactive element (either active or passive), a light animation is displayed to alert the user that a set of additional actions is available, selecting the given asset, which will be the target of such actions. Clicking on a target, another menu showing other actions available to the active character will be listed, and a button to make the target the

active character. A text box lets the user specify additional actions not listed in the menu. Moreover, the *talk to* action has been defined for actors to enable the user to detail a dialog, either typing it with the keyboard or registering it with a microphone, which will be included in the storyboard. When an action is selected, it is played in the 3D scene if the corresponding animation is available.

Each time the user acts, a preview of the corresponding action's description, based on the status of the FSMs, is displayed at the bottom of the interface. These descriptions will be used to automatically label the storyboard vignettes. Furthermore, the stack of events is depicted as a timeline at the top of the interface, with the timeline automatically moving on each time a vignette is generated. The scope of the timeline is to define if actions taking place in the scene occur simultaneously or at a consecutive time for a given vignette. Even if the events are queued in the stack, events with the same frame value are depicted as concurrent in the text description. If the user increments the timeline frame value, actions within the same scene will have different timings and be described sequentially. An undo button/shortcut lets the user undo the last event, resetting the character position if it is movement-related.

The camera can be selected as any other active character in the scene and positioned with a keyboard and mouse interface: the $WASD$ keys are used to pan the camera; the QE keys or the mouse wheel can be used to get closer or far away from the scene; tilting the mouse on its two axes rotate the camera with a standard look-at approach, as usually happen in 3D first-person video games. This approach enables the user to explore different camera angles and shots, even observing the scene from different actors' perspectives, as shown in Fig. 4. A slider on the right of the interface lets the user change the camera's focal distance.

Finally, a button/shortcut lets the user insert a save point in the stack, taking a screenshot of the scene based on the current view, adding a vignette to the storyboard, automatically generating the textual description, and adding the camera focal distance. Each vignette's tuple image/description is saved in an HTML template, and the textual description can be eventually edited.

FSM Editing and Creation. The user can edit or create the FSM of one of the available 3D assets through a 2D interface with two panels. The *status* panel provides a list of all the variable elements. Selecting one, a graphical representation of its FSM is provided in the form of a list, with each line containing: a starting status; an ending status; the action that activates the transition; and the actions available in the end status that do not generate a status change. This view also contains the transitions due to actions done by other actors to the selected one. The *actions* panel provides a list of actors and a list of targets, represented by the variable elements (both active and passive): a list of actions is displayed when selecting an actor and a target. This list represents the actions available to the actors if the target is in range of that action.

Fig. 4. Three different camera views for the same scene.

5 User Test

A 3D scene has been created with the proposed tool to evaluate the system's usability. The 3D models used to furnish the scene are freely available 3D models for non-commercial use downloaded from the Sketchfab website [20]. The 3D characters, with the rigging and animations, have been downloaded from Adobe Mixamo [21] and are also freely available for non-commercial projects.

5.1 Use Case

A script has been written based on the available 3D assets, detailing locations, actions, timing (for concurrent actions), and the camera shot. Eleven students from Politecnico di Torino have been involved in this study: 9 are master's degree students in Cinema and Media Engineering, and 2 are bachelor's degree students in Design and Communication. All the testers were volunteers with previous experience in storyboarding for university projects. The tests have been carried out on one user at a time to avoid learnability effects.

5.2 Test Procedure

Experiments were carried out adopting the following procedure:

1. the user is introduced to the project, the essential concepts, the scope of the tests, and the test procedure;
2. the user is guided through the user interface and the available functionalities;
3. the user is provided with a tutorial scene to freely test the different functionalities for a limited amount of time (10 to 15 min);
4. if the user is satisfied trying out the application, a new scene is loaded, and a paper script is provided to the user; the user starts creating the storyboard, and both times and errors are manually registered;
5. once the users complete the storyboard, they are asked to fill out the SUS questionnaire [22] to measure the system's usability;
6. finally, the user can provide additional feedback on the application usage by answering four open questions.

5.3 Tests' Results

All the users were able to complete the tests, with an average completion time of $7m10s$ and a standard deviation of $2m28s$. On average, one error and a half were committed by each user. Most of the errors involved the concept of simultaneous versus sequential actions, with the user not noticing or forgetting the script's indications when deciding the actions' timing. SUS scores are detailed in Table 1.

Table 1. System Usability Scale questionnaire results for the ten questions ($q_1 - q_{10}$) from eleven testers ($t_1 - t_{11}$) and the average (AVG) score.

	q1	q2	q3	q4	q5	q6	q7	q8	q9	q10	**SCORE**
t1	3	2	4	2	5	1	5	2	3	2	77.5
t2	3	2	5	2	5	1	4	2	4	1	82.5
t3	4	2	4	2	5	2	5	1	5	1	87.5
t4	2	1	5	2	5	1	3	1	5	2	82.5
t5	2	1	4	2	4	1	5	1	3	2	77.5
t6	3	2	4	1	3	3	5	2	4	1	75.0
t7	5	1	4	1	5	1	4	1	5	2	92.5
t8	5	2	5	2	5	1	5	2	5	1	92.5
t9	4	4	4	1	4	1	4	3	3	2	70.0
t10	4	2	4	2	4	1	4	2	3	1	77.5
t11	4	1	4	1	4	1	4	2	3	1	82.5
AVG											**81.6**

The open questions aimed at highlighting the most and least appreciated feature or functionality, suggestions for missing functionalities considered helpful by the user, and a personal comment comparing the proposed method with those usually adopted by the users.

5.4 Discussion and Limitations

Based on the feedback provided by the users at the end of the tests, the most appreciated feature was the overall 3D storyboarding approach and its easiness of use, from the scene creation to handling the characters and performing actions (5 out of 11); the automatic text generation and managing the characters actions with an FSM approach have been considered equally relevant (3 out of 11).

On the other hand, the most relevant difficulties involve camera usage and active character selection (6 out of 11). The first problem depends on the user background: those accustomed to video games did not have issues with the proposed interface; the others would have preferred a gimbal or other mechanics available in 3D modeling and 3D animation tools. This alternative commands to handle the camera could be easily added in a future version of the application.

Moreover, the scene view may result in characters obscured by other 3D assets and difficulties in selecting; thus, the user may waste much time changing the camera view to achieve the proper selection. Using the timeline to define the actions' chronological order, combined with the limitations of the undo action, was considered a significant problem by two users out of 11. In contrast, three users could not find any feature to disfavor. The user suggested further enhancing the undo feature, even if there are technical limitations due to the FSM approach, and despite the necessity to maintain overall consistency in the system (3 out of 11). Two users suggested adding more shortcuts to speed up the storyboarding process. Other suggestions comprehend: the automation of the camera shot setup, calculating the zoom level necessary for different shot options based on the camera-target distance; the visibility of the actions/events in the stack; the possibility of changing the duration of each scene at the end of the storyboarding phase; finally, adding more 3D assets to create the scenes. Pertaining a subjective comparison with the testers' traditional techniques previously used for storyboarding, 6 out of 11 users believe that the proposed approach is easier and faster than a sketch-based approach, especially for people not accustomed to drawing. However, two users think drawing is a more flexible and creative approach since the assets' availability does not limit the user. Two users could not choose between the proposed method and the existing ones.

Despite these problems, the questionnaires' results show that the proposed system was highly appreciated by the user. Comparing the results with the SUS chart, five users out of eleven obtained a good score (range $68-80.3$), whereas six users obtained a higher score (≥ 80.3). The average SUS score of 81.6 suggests that the users believe that the proposed system provides excellent usability.

6 Conclusion and Future Work

This research proposes a 3D storyboarding tool to improve existing storytelling approaches. A novel storyboarding pipeline is proposed, which can automatically generate a storyboard including camera details and a textual description of the events occurring in each scene. Users create storyboards by selecting actors, performing available actions, positioning the camera in the 3D scene, and taking screenshots to save a vignette of the storyboard; the corresponding description is generated based on the actors' actions and their status. A software implementation of the proposed pipeline has also been developed in the guise of a 3D desktop application aimed at expert and novice storyboarders. The system has been tested to evaluate its usability. Preliminary results confirm that the users have appreciated the application.

Future works will aim to improve the user interface, exploiting the feedback provided by the testers. Apart from enhancing the camera handling, a further option is to provide a dropdown menu to select a specific character directly and key shortcuts to move to the next/previous actor available in the scene. Despite the necessity to guarantee system consistency, the editing feature could be improved, and the stack of actions could be made visible to the user. The

automatic text generation can be further enhanced with an AI approach, such as ChatGPT [17]. The scene creation phase can be enriched with more 3D assets and animations for the variable elements.

References

1. Photoshop homepage. https://www.adobe.com/products/photoshop.html. Accessed 31 Mar 2023
2. Storyboardthat homepage. https://www.storyboardthat.com. Accessed 31 Mar 2023
3. Canva homepage. https://www.canva.com. Accessed 31 Mar 2023
4. FrameForge homepage. https://www.frameforge.com. Accessed 31 Mar 2023
5. ShotPro homepage. https://www.shotprofessional.com. Accessed 31 Mar 2023
6. Gouvatsos, A. 3D storyboarding for modern animation. Doctoral dissertation, Bournemouth University (2018)
7. Gouvatsos, A., Xiao, Z., Marsden, N., Zhang, J.J.: Automatic 3D Posing from 2D Hand-Drawn Sketches. In: PG (2014)
8. Pizzi, D., Lugrin, J.L., Whittaker, A., Cavazza, M.: Automatic generation of game level solutions as storyboards. IEEE Trans. Comput. Intell. AI Games **2**(3), 149–161 (2010)
9. Lino, C., Christie, M., Ranon, R., Bares, W.: The director's lens: an intelligent assistant for virtual cinematography. In: Proceedings of the 19th ACM iNternational Conference on Multimedia, pp. 323–332 (2011)
10. Kapadia, M., Frey, S., Shoulson, A., Sumner, R. W., Gross, M.H.: CANVAS: computer-assisted narrative animation synthesis. In: Symposium on Computer Animation, pp. 199–209 (2016)
11. Louarn, A., Christie, M., Lamarche, F.: Automated staging for virtual cinematography. In: Proceedings of the 11th ACM SIGGRAPH Conference on Motion, Interaction and Games, pp. 1–10, November 2018
12. Ronfard, R., Gandhi, V., Boiron, L., Murukutla, A.: The prose storyboard language: a tool for annotating and directing movies (version 2.0, revised and illustrated edition). In: Eurographics Workshop on Intelligent Cinematography and Editing, vol. 4 (2022)
13. Kim, H., Ali, G., Hwang, J.I.: ASAP: auto-generating storyboard and previz with virtual humans. In: 2021 IEEE International Symposium on Mixed and Augmented Reality Adjunct (ISMAR-Adjunct), pp. 316–320. IEEE, October 2021
14. Evin, I., Hämäläinen, P., Guckelsberger, C.: Cine-AI: Generating Video Game Cutscenes in the Style of Human Directors (2022). arXiv preprint arXiv:2208.05701
15. Unity homepage. https://unity.com. Accessed 31 Mar 2023
16. Chen, W., Liu, X., Niu, J.: SentiStory: a multi-layered sentiment-aware generative model for visual storytelling. IEEE Trans. Circuits Syst. Video Technol. **32**(11), 8051–8064 (2022)
17. ChatGPT homepage. https://openai.com/blog/chatgpt. Accessed 31 Mar 2023
18. WordNet homepage. https://wordnet.princeton.edu. Accessed 31 Mar 2023
19. Syn WordNet homepage. https://developer.syn.co.in/note/wordnet/release-notes.html. Accessed 31 Mar 2023
20. Sketchfab homepage. https://sketchfab.com/feed. Accessed 31 Mar 2023
21. Adobe Mixamo homepage. https://www.mixamo.com/. Accessed 31 Mar 2023
22. Brooke, J.: SUS-a quick and dirty usability scale. Usabil. Eval. Indus. **189**(194), 4–7 (1996)