January 19, 2017

Smooth Fly-to-Orbiting Animation for Viewing Points of Interest in 3D

Jingyi Fang

Sean Askay

Smooth Fly-to-Orbiting Animation for Viewing Points of Interest in 3D

Introduction

The present disclosure provides systems and methods for providing a smooth fly-to orbiting animation within a user interface. In particular, mapping service providers have collected and reconstructed large amounts of three-dimensional (3D) mesh and pixel data for many cities over the world, mountains, and other geographic areas. In addition, many three-dimensional models have been built to reflect manmade and natural structures. To allow users to experience such geographic areas, it is important to present those three-dimensional contents in an intuitive and immersive way via a geographic user interface.

Due to the difficulty in performing heading and tilt manipulation within a user interface, a significant amount of users are unable to experience the three-dimensional contents. For example, a user may search for a point of interest (POI) within a geographic user interface. The user interface may "fly" the user to view that point of interest. However, current fly animations may provide the user with a static, top-down, 0-heading, 0-tilt view. While such a view may involve less processing resources (e.g., a top-down, zero tilt view encompass less geometry and textures for the rendering system to handle per frame), such a view can make it difficult for a user to experience (and the user interface to show) three-dimensional content.

Summary

The present disclosure proposes to solve the challenges described above by introducing a programmatic smooth fly-to-orbiting animation generation system. More particularly, one aspect of the present disclosure provides a smooth fly-to-orbiting animation that will be provided to a user via a user interface such that a user can more readily view the 3D content associated with a point of interest.

In particular, a user device (e.g., a laptop computer, mobile phone, or the like) can present a user interface for a user of the user device. The user interface can include, for example, a rendering of a three-dimensional map interface. The user interface can be associated with a virtual camera that allows a user to navigate and control the view presented in the user interface. For example, a user can navigate through a three-dimensional environment (e.g., associated with a substantially spherical three-dimensional model of the Earth) presented by the user interface to display three-dimensional data (e.g., rendered using satellite and/or other imagery). The three-dimensional environment can include a virtual camera that defines what three-dimensional data to display. The virtual camera can have a perspective according to its position and/or orientation. By changing the position, orientation, zoom, etc. of the virtual camera, a user can navigate through the three-dimensional environment presented by the user interface.

The user device can present an animated motion of the virtual camera according to the present disclosure. For instance, a user can provide user input indicating a point of interest (e.g., a skyscraper in City A). The user device (and/or a server system associated with the geographic user interface) can identify a current view of the virtual camera and determine a proper final view of the point of interest. An algorithm can be used to generate a smooth camera transition animation that flies the virtual camera from the current view to the final view and at the same time starts an orbiting animation around the point of interest. Accordingly, the virtual camera of the user interface can navigate the user such that the user can better experience the three-dimensional content associated with the point of interest (e.g., the skyscraper).

Detailed Description

Figure 1 depicts an example computing system 100 for providing a smooth fly-to-orbiting animation using the processes described herein. The system 100 can include a provider

computing system 102 and one or more user device(s) 104. The provider system 102 and the user device(s) 104 can be interconnected via a direct connection and/or can be coupled via one or more communications network(s) 106, such as a LAN, WAN, the Internet, etc., which may be wired and/or wireless and/or can include any number of wired and/or wireless communication links.

The provider system 102 can be associated with, for example, a mapping service provider and/or another entity that provides data to be rendered in a user interface. In some implementations, the provider system 102 can be associated with a three-dimensional body to be displayed (e.g., architect, building owner, government entity). The provider system 102 can include various components for performing various operations and functions. For instance, the provider system 102 can include one or more processor(s) and one or more memory device(s). The one or more memory device(s) can store instructions that when executed by the one or more processor(s) cause the one or more processor(s) to perform operations and functions. The provider system 102 can be coupled to and/or can include various databases, such as a database 108. The database 108 can include data (e.g., geographic data) that is accessible by the provider system 102. The provider system 102 can provide various data from the database 108 to the user device(s) 104 via the one or more network(s) 106.

The user device(s) 104 can be various types of user devices, such as, a tablet, a personal digital assistant (PDA), a laptop computer, a desktop computer, a phone, a smart phone, a computerized watch (e.g., a smart watch), other types of wearable computing devices, and/or any other type of mobile and/or non-mobile computing device. The user device(s) 104 can include various components for performing various operations and functions as described herein. For instance, a user device 104 can include a display device 109 and one or more computing

device(s) 110. The display device 109 can include an output device configured to display a user interface, such as a CRT, LCD, plasma screen, touch screen, etc. The computing device(s) 110 can include one or more processor(s) and one or more memory device(s). The one or more memory device(s) can store instructions that when executed by the one or more processor(s) cause the one or more processor(s) to perform operations and functions.

The computing device(s) 110 can include various components for implementing a user interface. The computing device(s) 110 can include a camera component 112 (e.g., camera manipulation system) that is configured to receive data indicative of a user input (e.g., raw input data) from a user and generate an output based, at least in part on the input data. For example, the camera component 112 can be associated with a virtual camera 114 that can be used to navigate within a user interface. The virtual camera 114 can have a perspective according to its position and/or orientation. By changing the position, orientation, zoom, etc. of the virtual camera 114, a user can navigate through an environment presented by a user interface. By way of example, the user interface can be a geographic map interface that presents a three-dimensional environment. In some implementations, the three-dimensional environment can include a substantially spherical three-dimensional model of the Earth (e.g., rendered using satellite imagery and/or other imagery). The camera component 112 can receive data indicative of a user input directed to causing a motion of the virtual camera 114 so that the user can navigate the environment of the user interface (e.g., to see different views, orientations, positions, locations). The camera component 112 can process the data indicative of the user input and generate an output based, at least in part, on that data. The output can be indicative of one or more conditions(s) that define the camera (e.g., location conditions, orientation conditions). In some implementations, the output can be indicative of an animation that includes

a plurality of location and orientation conditions according to which the virtual camera 114 is to navigate. The camera component 112 can provide the output to a rendering component 116 of the computing device(s) 110.

The rendering component 116 can be configured to receive the output 116 and render a frame of the user interface based, at least in part, on the output 116. For example, the rendering component 116 can obtain data 118 (e.g., geographic mapping data, geometries, textures) from the provider system 102 based, at least in part, on the output, which can indicate the view, orientation, position, location, etc. of the virtual camera 114 to be presented in the user interface. The rendering component 116 can render a frame of the user interface based, at least in part, on the data 118. In this way, the virtual camera 114 can be manipulated to view, for example, the spherical three-dimensional model of the Earth from different perspectives.

The virtual camera 114 can navigate a user interface according to an animation defined by a CameraSource object. A camera view (of the virtual camera 114) can be based, at least in part, on a LookAtCamera object, which can be mathematically defined by a plurality of parameters. Those parameter(s) can include, for instance, the location (e.g., latitude, longitude, altitude) of a point the camera is looking at (e.g., a focus point), the distance from the focus point to the camera's position, the heading, tilt, and roll of the camera with respect to the focus point, etc. A camera animation can be generated by creating CameraSource objects that can be read by a CameraAnimationPlayer at each simulation time to retrieve a LookAt Camera and set for the rendering component 116 to render a frame. Under the present disclosure, an algorithm can be used to compose Camera Source objects that can mathematically produce an interpolation result for a smooth fly-to-orbit animation.

Figure 2 depicts an example diagram 200 for creating a BlendingCameraSource object 201 and to implement the smooth-fly-to-orbit animation of the present disclosure. Such process can be implemented using an algorithm performed via the user device 104 and/or another computing system associated with the user interface (e.g., such as a web-based server providing the user interface via an application).

The BlendingCameraSource object 201 takes two CameraSource objects and blends them for their overlapping timeline. Such blending can include a Fly Animation CameraSource 202 and an Orbiting Animation CameraSource 203. A Fly Animation CameraSource 202 can be initialized with, for example, a start LookAtCamera, a final LookAtCamera, a fly duration and a flight path type (e.g., linear, parabolic, etc.). The CameraSource object internally has an interpolation algorithm that produce a LookAtCamera given any input interpolation parameter t (0~1). An Orbiting Animation CameraSource 203 can be initialized with, for example, a starting LookAtCamera of the point of interest, total rounds of orbiting (e.g., floating points, such as 0.34 rounds of orbiting, negative to rotate counter-clockwise), and a duration of the orbiting animation. When it is played, the Orbiting Animation CameraSource 203 increments the heading of the LookAtCamera given the current interpolation time (e.g., 0 to 1).

In some implementations, to adjust the nature of the orbiting portion of the animation, the Orbiting Animation CameraSource 203 itself can be a combination (e.g., a blended CameraSource) of two or more CameraSources. For instance, the Orbiting Animation CameraSource 203 can include a fast but short orbiting animation CameraSource (e.g., ~ 0.2 rounds about the point of interest) and a fast but short orbiting animation CameraSource (e.g., ~ 0.8 rounds about the point of interest). This blending can make the orbiting animation start faster to better match the end of the fly animation.

The BlendingCameraSource 201 blends the Fly Animation CameraSource 202 and an Orbiting Animation CameraSource 203. For instance, the BlendingCameraSource 201 identifies an overlap time period between the Fly Animation CameraSource 202 and an Orbiting Animation CameraSource 203 and blends them (e.g., using linear interpolation, linear combination) over that period to produce a smooth transition animation from the fly animation to the orbiting animation. As a result of the blending, the BlendingCameraSource 201 can have three stages: the first stage is the output of the first CameraSource's animation (e.g., the Fly Animation CameraSource 202), the second stage is a blending between the two input CameraSources, and the third stage is the output of the second CameraSource's animation (e.g., the Orbiting Animation CameraSource 203). An interpolation time associated with the blending can be calculated as: t = min(1.0, (current_time - animation_start_time) / animation_duration). In some implementations, one or more easing function(s) can be applied to the interpolation time "t". The easing functions can map "t" from (0-1) to (0-1) in a continuous manner. Additionally, or alternatively, the easing function(s) can turn a straight line into a curvature. Example easing function(s) can include cubic ease, exponential ease, etc.

In some implementations, a common heading orientation of the virtual camera 114 can be used to further ease the transition between the two CameraSources 202, 203. For example, the user device 104 (and/or other computing system) can first read from the Fly Animation CameraSource 202 to determine an animation heading orientation (e.g., clockwise, counter-clockwise) associated with the virtual camera 114 during the fly animation. The user device 104 (and/or other computing system) can then use the animation heading orientation to set the direction for the orbiting animation of the Orbiting Animation CameraSource 203. In this way,

the fly-to-orbiting animation can have a smooth transition in heading rotation as it transitions between CameraSources.

In some implementations, additional user interface elements can be provided while the virtual camera 114 is navigating according to the smooth fly-to-orbit animation. For example, an interactive 3D orbiting icon can be presented via the user interface when the virtual camera 114 is orbiting the point-of-interest. The user can interact with the icon (e.g., via a mouse click) to stop, re-start, adjust the speed, position, orientation, etc. of the orbiting animation. In some implementations, the user can perform any kind of view manipulation to stop the orbiting immediately. When the user adjusts the view and interacts with the icon to re-start the orbiting animation, the view presented in the user interface will smoothly transition to start orbiting at a certain tilt value and the orbiting animation will use the current zoom level as the zoom level when orbiting about a point of interest. In this way, the animation can still provide a smooth presentation of a point of interest despite user interruption of the animation.

Figures

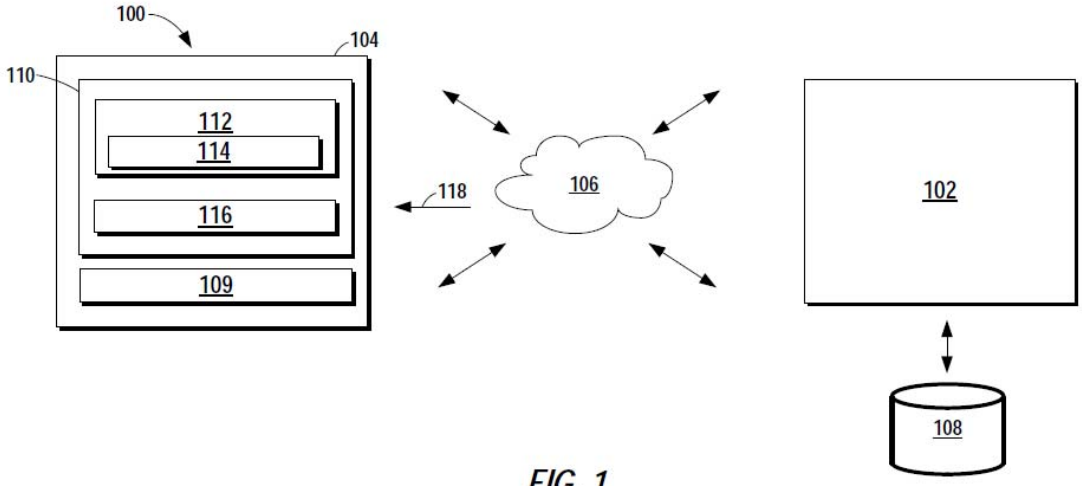

FIG. 1

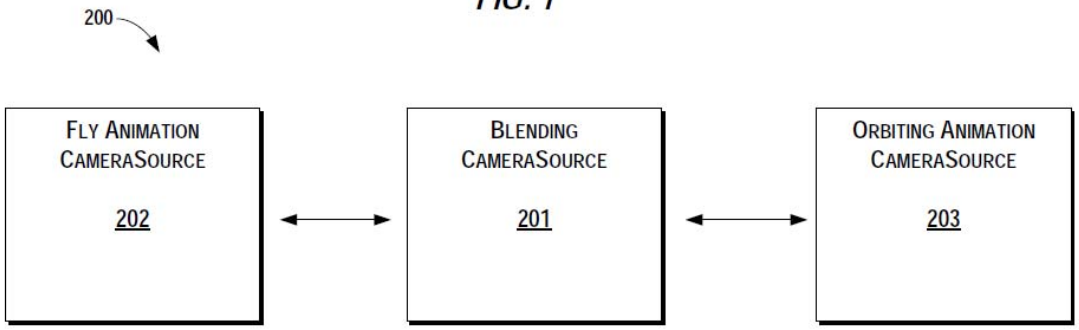

FIG. 2

Abstract

The present disclosure describes systems and methods that combine two or more CameraSource objects to create a smooth fly-to-orbit animation within a three-dimensional user interface. More particularly, an algorithm is used to blend a fly animation CameraSource with and an orbiting animation CameraSource over an interpolation time period to create a CameraSource object that allows a smooth, user-friendly transition between the animations for use in a user interface (e.g., presenting three-dimensional data). Keywords associated with the present disclosure include: mapping; user interface; three-dimensional; point of interest; geographic area; virtual camera; navigation; camera source; animation; fly; orbit, blending; transition; interpolation.