Storytelling in the Metaverse: From Desktop to Immersive Virtual Reality Storyboarding

Storytelling in the Metaverse: from desktop to immersive virtual reality storyboarding

1st Federico Manuri
DAUIN
Politecnico di Torino
Torino, Italy
federico.manuri@polito.it

2rd Andrea Sanna
DAUIN
Politecnico di Torino
Torino, Italy
andrea.sanna@polito.it

3th Francesco De Pace
Institute of Visual Computing and Human-Centered Technology
TU Wien
Vienna, Austria
francesco.pace@tuwien.ac.at

***Abstract*—Creatives from the animation and film industries have always been experimenting with innovative tools and methodologies to improve the creation of prototypes of their visual sequences before bringing them to life. In recent years, as realistic real-time rendering techniques have emerged, the increasing popularity of virtual reality (VR) can lead to new approaches and solutions, leveraging the immersive and interactive features provided by 3D immersive experiences. A 3D desktop application and a novel storyboarding pipeline, which can automatically generate a storyboard including camera details and a textual description of the actions performed in three-dimensional environments, have already been investigated in previous work. The aim was to exploit new technologies to improve existing 3D storytelling approaches, thus providing a software solution for expert and novice storyboarders. This research investigates 3D storyboarding in immersive virtual reality (IVR) to move toward a new storyboarding paradigm. IVR systems provide peculiarities such as body-controlled exploration of the 3D scene and a head-dependant camera view that can extend features of traditional storyboarding tools. The proposed system enables users to set up the virtual stage, adding elements to the scene and exploring the environment as they build it. After that, users can select the available characters or the camera, control them in first person, position them in the scene, and perform actions selecting from a list of options, each paired with a corresponding animation. Relying on the concept of state-machine, the system can automatically generate the list of available actions depending on the context. Finally, the descriptions for each storyboard panel are automatically generated based on the history of activities performed. The proposed application maintains all the functionalities of the desktop version and can be effectively used to create storyboards in immersive virtual environments.**

***Index Terms*—Tabletop storyboarding, immersive virtual reality, character animation, authoring tool.**

I. Introduction

Creatives from the animation and film industries have always been experimenting with innovative tools and methodologies to improve the creation of prototypes of their visual sequences before bringing them to life. From traditional methods such as hand-drawn storyboards and physical mockups, information technology introduced sketch-based and picture-based 2D drawing applications. In recent years, as realistic real-time rendering techniques have emerged, 3D modeling and animation tools have been researched and explored to support storyboarding and movie prototyping. In this context, the increasing popularity of virtual reality (VR) can lead to new approaches and methodologies, leveraging the immersive and interactive features provided by 3D VR experiences. Some research studies explored the usage of VR in the animation and film domains. Penrose Studios investigated content creation for VR ad Augmented Reality (AR) devices, researching tools that could improve the traditional animation pipeline [1]. In [2], Herikson et al. investigated a workflow tailored to the specific needs of professionals creating storyboards for VR films, and a multi-device (tablet and head-mounted display) storyboard tool has been proposed. Other researchers focused on using VR technology as a drawing and authoring tool, developing applications to create artistic content, such as Oculus Story Studio [6] or TiltBrush [5]; the tracked VR controllers are used as brushes to paint in the 3D environment. More recently, Vogel et al. developed a Unity plugin for interactively creating animations in VR [3], whereas Galvane et al. focused on exploiting VR for the previsualization stage [4]. Renowned commercial solutions, such as Frameforge 3D [7] or ShotPro [8], are usually complex to master and present a steep learning curve.

II. Description of the contribution

This research aims to investigate the usage of an immersive virtual reality (IVR) storyboarding tool. A 3D desktop application and a novel storyboarding pipeline, which can automatically generate a storyboard with camera details and a textual description of the actions performed in the virtual scene, have been presented in previous work [9]. The aim was to exploit new technologies to improve existing 3D storytelling approaches, thus providing a software solution for expert and novice storyboarders. However, the proposed tool presents a challenging learning curve from a usability point of view. Despite the possibilities of setting up and posing characters in a 3D environment, camera positioning is still a complex task that requires previous knowledge from other software, such as 3D modeling or 3D animation software, or a video gamer background (e.g., first-person shooter video games). This research investigates the usage of IVR to improve the usability of the system mentioned above. The user interface has been completely redesigned to fit the VR context: introducing dedicated controllers instead of a mouse and keyboard required to rethink the user interaction paradigm, thus investigating a

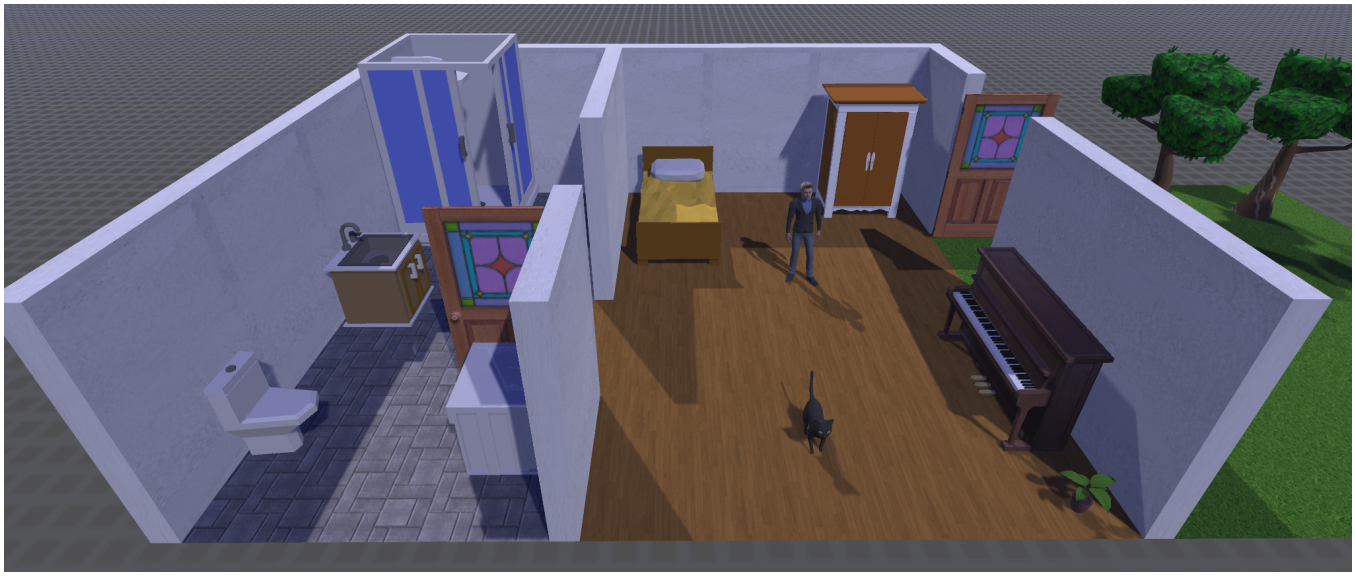

Fig. 1. A sample virtual stage created with the proposed system.

different way to provide users the same functionalities of the desktop version. The system enables the user to set up the virtual stage, adding elements to the scene by choosing among environmental tiles, objects, and characters (Fig. 1). After that, users can select the available characters or the camera, control them in first person, position them in the scene, and perform actions from a set of possibilities, each paired with a corresponding animation. To this end, an IVR can enable users to control 3D actors from a first-person point of view with a greater sense of presence and identification than desktop applications. The list of available actions is generated relying on the concept of state-machine, with each state depending on the context: the position in the virtual world, the proximity with other objects or characters, and the record of actions previously performed. This way, the system can automatically generate the descriptions for each storyboard panel based on this state-machine approach and the activities' history .

III. System Design

Based on the functionalities developed for the desktop storyboarding application and the analysis of the state of the art, a list of possible functionalities has been defined. These functionalities have been evaluated with the MoSCoW method, a prioritization technique developed by Dai Clegg [10] in 1994 for use in rapid application development (RAD) and extensively used from 2002 with the dynamic systems development method (DSDM) [11]. This method, also known as MoSCoW prioritization or MoSCoW analysis, is used in business analysis, project management, and software development to reach a common understanding with stakeholders on the importance they place on delivering each requirement. MoSCoW is an acronym derived from the first letter of each of four prioritization categories: M - Must have, S - Should have, C - Could have, W - Won't have. The MoSCoW analysis for this research has involved students from the master's degree in Cinema and Media Engineering of Politecnico di Torino. These students learn advanced skills in cinema and media technologies and languages to manage the innovation processes of new digital production companies. Thus, they represent real users that may employ the proposed system for their working activities in the future. Questions on storyboarding and virtual reality were posed to evaluate the students' experience and proficiency in the proposed research domain. This step was necessary to identify and remove outliers represented by students who claimed or proved no knowledge of storyboarding.

A. MoSCoW analysis discussion

Thirty-five students participated in the MoSCoW analysis. Table I shows the obtained results: each category's percentage and priority values for each requirement are displayed. Percentages have been used to sort the requirements in each category. The MoSCoW analysis results have been combined with the requirements already developed for the desktop version of the proposed storyboarding application. As a result, some of the requirements have already been developed and only need to be ported into the VR system. Other criteria were partially developed or need further development to match the users' demands. Finally, some requirements need to be developed from scratch.

This analysis allowed us to define a shorter set of requirements for the proposed project, which resulted in the following list of Must requirements:

1) improving the system usability making it easier to select actors' actions and poses, undo actions, and save the system status;
2) adding multiple cameras to the scene, switching among them, and saving their parameters (focal length, sensor type, etc.);
3) adding virtual lights to the scene by manually editing their parameters or loading presets.

TABLE I
REQUIREMENTS PRIORITIZATION WITH THE MOSCOW METHOD.

#	Requirements	M%	S%	C%	W%	Result
1	Insert objects and characters into the scene, choosing them from a list, and define their position and orientation;	0.94	0.06	0.00	0.00	M
2	Make it possible to view the storyboard once completed	0.89	0.11	0.00	0.00	M
3	The application must allow the addition and movement of virtual cameras to frame the scene from different points in space and take the screenshots that make up the storyboard vignettes	0.86	0.14	0.00	0.00	M
4	Removing objects or characters from a scene	0.83	0.09	0.09	0.00	M
5	Create screenshots of the scene that represent the storyboard panels	0.74	0.23	0.03	0.00	M
6	Allowing you to view what is framed by a virtual camera within the application with a real-time preview	0.69	0.26	0.03	0.03	M
7	Possibility of being able to save a created scene and be able to reload it at a later time	0.69	0.29	0.03	0.00	M
8	Provide one or more UNDO levels to return to the condition of the ”previous screenshot”	0.66	0.20	0.14	0.00	M
9	Edit the final storyboard by rearranging the images and changing the descriptions	0.63	0.26	0.09	0.03	M
10	For each camera, it must be possible to set the parameters (focal length, sensor type, etc.)	0.51	0.37	0.09	0.03	M
11	Possibility for the user to add indications within the shot that specify the movements of the camera and of the characters (e.g. Arrows that indicate the directions of movement, zoom in/out ...)	0.51	0.31	0.14	0.03	M
12	Giving the possibility to define the duration of an action or a shot	0.43	0.37	0.17	0.03	M
13	Possibility to define the poses of the characters manually, modifying the armor of the model	0.43	0.31	0.17	0.09	M
14	Possibility of inserting virtual lights into the scene	0.43	0.40	0.06	0.11	M
15	The system allows the user maximum freedom to be able to choose any action among those available for the characters to perform at any time regardless of the context	0.40	0.29	0.29	0.03	M
16	The application must support a gestural interface	0.34	0.29	0.26	0.11	M
17	Make it possible to view the storyboard before completion	0.43	0.43	0.14	0.00	S
18	Modify the parameters of the lights (power, area, color)	0.34	0.43	0.17	0.06	S
19	Rename the characters and objects in the scene	0.29	0.46	0.14	0.11	S
20	Possibility of interaction between a character and the other objects present in the scene	0.26	0.43	0.20	0.11	S
21	Possibility to define the duration of an action represented by a vignette of the storyboard	0.26	0.43	0.31	0.00	S
22	Possibility of animating a character by applying predefined poses to the model, which can be selected from a library	0.23	0.51	0.17	0.09	S
23	Possibility of interaction between character and character	0.20	0.34	0.37	0.09	S
24	The user must have the ability to view and edit the scene in different scales (tabletop, 1:1 scale in first person)	0.11	0.54	0.29	0.06	S
25	Possibility to have a lighting preset for outdoor environments	0.11	0.46	0.40	0.03	S
26	Create a preview of the actions performed in the application through a video	0.06	0.46	0.31	0.17	S
27	Possibility of controlling the characters in the scene by moving them in space and animating them with commands, like in a video game	0.14	0.31	0.46	0.09	C
28	The system must verify the temporal consistency of the actions between consecutive cartoons	0.11	0.34	0.34	0.20	C
29	Based on the context, the system allows you to select only certain actions that are consistent with it	0.11	0.31	0.29	0.29	C
30	Have a description, automatically generated, that accompanies each shot of the storyboard	0.06	0.20	0.51	0.23	C
31	The application must support a voice interface	0.06	0.09	0.46	0.40	C

Another relevant requirement regards making the storyboard easier to edit and adding more details to the storyboard's vignettes. However, whereas camera and light parameters can be automatically added to the vignettes' descriptions, further storyboard editing can be easily performed with a 2D desktop interface (e.g., changing the vignettes' order). Thus, this feature has not been investigated in this paper.

B. Interface

Porting the 3D storyboard application to a virtual reality environment requires changes to the user interface to translate the actions mapped on the mouse and keyboard into easy interactions with the hand controllers. Since the aim is to provide a system independent from the VR headset, some default design rules for interacting with the UI are defined as follows, assuming a trackable headset in a given 3D physical space and controllers for both hands:

1) mouse interaction with the element in the scene is provided through raycasting from the controllers; different types of selection may be enabled based on the number or type of buttons pressed;
2) camera preview, in terms of movements and rotations, is replaced by the user's physical movement in the 3D environment, measured by headset tracking;
3) static menus from the desktop UI are replaced by dynamic menus that can be opened through buttons on the physical controllers or upon item interaction and appears as 2D panels in front of the user in the 3D world;
4) the user can interact with the UI elements by collision or by raycasting.

C. Finite State Machine

The desktop storyboard application presents a constrained approach based on a finite state machine (FSM) strategy. 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), with nodes representing different statuses and transitions representing actions available based on the current status. The status of passive elements (e.g., a switch or a door) depends solely on the actions performed by

other active elements in the scene. Instead, the status of active elements, such as characters, may depend on three factors:

- actions performed by the characters that changed their previous status;
- their position in the 3D space: based on their proximity with other variable elements, additional actions may be available to them;
- their previous interactions with other variable elements: e.g., if the character picked up an object, additional actions may be available.

However, one of the Must requirements is to allow the user maximum freedom to choose any action for the characters to perform at any time, regardless of context. This is probably because the traditional approach to storyboarding does not require performing every action that the actor would execute on the scene but simply drawing the desired action in the vignette and writing the corresponding description. Thus, moving from one situation to another is possible without worrying about the system's consistency. However, this approach may lead to an inconsistent state of the FSMs of the variable elements, and the system may not be able to automatically generate a proper description based on the actions performed by the user. To this end, the FSM approach proposed in the desktop application has been improved: a recording button has been added to allow the user to choose when to record the actions. When the recording is on, the consistency of the actions is guaranteed. When the recording is off, the user can either perform actions in the scene or change the elements' status by editing the scene. When a new status for the variable elements has been defined, users can switch back to storyboarding, and only the starting status will be considered for what pertains to the FSM consistency upon activating the recording of the actions.

Another consistency problem is related to allowing users to edit the scene whenever they want. If they introduce new elements in the scene, only the nearby or holding properties of active elements would be affected. However, removing an element from the scene may affect the status of a variable element. Let's assume, for example, that a character is sitting in a car in the first vignette, and the car is removed in the second vignette. The character status is preserved if the character is moved out from the car before deleting it. Otherwise, if the car is deleted with the character still inside, the system cannot define the character's status. Thus, when the character status depends on an object, and removing the object and maintaining its FSM consistency is impossible, the character and the object define a group and are removed together.

Finally, a stack has been used in the desktop application to save the actions performed by the characters. This allows the user to save the chain of events in a single vignette of the storyboard, simply generating a save-point in the stack; then, all the events in the stack following the last save-point should be converted into sentences to label the vignette automatically. However, to introduce the concept of 'undo' and roll back on the user's actions, it is necessary to save an actor's coordinates every time a movement action is started. On the other hand, rolling back on actions implies rolling back both the actor's action and, eventually, the object involved in the interaction (either an item or a character). An action may provoke a transition to a different state, and the system should check all the entities involved. Thus, rolling back on the action stack will allow users to cancel wrong actions and, eventually, even the current vignette up to the previous save point.

IV. System Implementation

The design requirements described in the previous section guided the development of an immersive virtual reality application based on the desktop prototype. Unity 3D[1] has been chosen to deploy the system since it is a 3D game engine freely available for research and highly compatible with various virtual reality headsets.

A. System Architecture

Unity 3D and the SteamVR[2] plugin have been used to deploy a 3D application compatible with the most recent head-mounted displays, such as the HTC VIVE or the Oculus Rift. The HTC VIVE Pro[3] has been chosen for developing the proposed application since it is one of the most performing devices on the market regarding technical specifications. Using the Lighthouse technology, two infrared wireless cameras (the HTC Base Stations) can track the headset and the controllers in a room-scale area of up to 4.5 x 4.5 meters. Additional HTC Sensors can be employed to extend the working space further. The HTC VIVE Pro system includes two HTC VIVE controllers to provide easy interaction with the virtual objects. These controllers track the user's hands' position in the virtual scene and provide a physical interface with buttons, triggers, and touchpad surfaces. The SteamVR plugin provides an easy way to map all the actions defined in the virtual reality application to the HTC VIVE controllers. Moreover, a virtual counterpart of the controllers is displayed in the virtual environment, thus enhancing the user's sense of presence. Visual feedback is provided on the virtual controller to mimic the interaction performed by the user on the real ones.

B. Pipeline

When the desktop application starts, it loads both 3D assets and (if available) their FSMs and animations into the system. Then, it provides three choices to the user:

- to create or edit an existing 3D scene;
- to start storyboarding from an existing 3D scene;
- to edit or create FSMs for the available 3D assets.

However, to simplify the proposed approach to storyboarding, allowing users to edit the scene whenever they want would be easier. When the application starts, the user can create a novel storyboard from scratch or load a previous project. When the 3D environment loading is complete, the user can freely switch between the editing and storyboarding views. Switching

[1]https://unity.com
[2]https://valvesoftware.github.io/steamvr_unity_plugin/
[3]https://www.vive.com/us/product/vive-pro-full-kit/

Fig. 2. An example of the UI panels available to select the actions based on the active actor (the woman) and the target (the woman herself on the left, the man on the right).

between the two views only affects the functionalities available to the users.

C. User Interface

Users may select active elements in the scene by simply pointing them with the raycasting emitted from the VIVE controller and pressing the trigger button. Then, a panel appears before the user, containing a list of available actions based on the context. Fig. 2 shows an example of the UI panels available to select the actions based on the active actor (e.g., the woman) and the target (the woman herself on the left, the man on the right). Selecting an active element by pressing the left grip button instead of the trigger button allows users to embody the chosen element: this will enable users to experience the actors' viewpoint and control their position and orientation by simply moving in the 3D environment. Moreover, when a character is selected, the user can

- select a point on the scene floor using the raycasting approach and the trigger button to make the character move to that point;
- select a passive or active element using the raycasting approach and the left grip button to open a panel with the list of available actions that the current element can perform on the target element;
- select an active element using the raycasting approach and the trigger button to change the current element and open the panel with its available actions;
- select an active element using the raycasting approach and the right grip button to embody the target element.

Once users choose an action for the current actor, if the recording is active, a play icon appears over the actor, and the animation associated with that action is played. Using the trackpad on the right controller, users can:

- set the default pose by pressing up;
- play and pause the animation by pressing down;
- move among the animation's frames by pressing left or right (only if the animation is paused) to select the preferred frame.

This allows the user to choose the most representative pose for the actor to be depicted in the storyboard vignette.

Users can switch between the storyboarding and editing views by pressing the system button. In the editing view, pressing the left grip button opens the panel with the list of 3D elements that can be added to the scene. Once selected, the user can define the element's starting position using the raycasting approach to point into the 3d environment, see a preview of the element, and push the trigger button to confirm the position. Elements can then be selected using the raycasting approach and the trigger button. Once selected, the element's position can also be changed using the trackpad and the grip buttons for fine adjustments.

Moreover, the system allows users to add cameras to the scene. Users can set a camera corresponding to their current view by pressing the left grip button. Cameras can also be added to the scene as any other 3D active element. Cameras are not rendered in the screenshots. Selecting a camera using the raycasting approach and pushing the left grip button opens a panel to set its parameters. Users may select a camera using the raycasting approach and the right grip button to embody the camera. Pressing the right grip button opens a panel in front of the user to show a scene preview from the available cameras. Selecting a camera from the panel changes the camera to be used to take the shot to generate the vignette.

Furthermore, the system allows users to add lights to the scene, like any other 3D passive element. Selecting a light using the raycasting approach and pushing the left grip button opens a panel to set its parameters. Selecting one or more lights using the raycasting approach and pressing the trigger button creates a group of lights that can be saved as a preset by pushing the right grip button. The preset contains the lights' type, position, orientation, and parameters. Presets are then made available in the add object menu.

Finally, the system allows opening a panel in front of the user displaying all the vignettes generated till that moment and the corresponding description, as per requirement number 2. Fig. 3 shows a sample shot from a storyboard created with the proposed system.

Fig. 3. A sample shot from a storyboard created with the proposed system.

V. CONCLUSIONS AND FUTURE WORKS

The IVR version of our original 3D storyboarding desktop tool has been developed with a completely redesigned user interface to fit the VR-dedicated hardware. The proposed application maintains all the functionalities of the desktop version and can be effectively used to create storyboards in IVR. User tests will be performed to compare the IVR and desktop versions of the proposed system. The comparison will consider the system usability, the workload, the number of errors, and the duration of the storyboarding process. A comparison with a traditional storyboarding system is another aspect to investigate in the future. However, many types of conventional storyboarding options are available: hand-drawn storyboards, physical mockups, and sketch-based or picture-based drawing applications. Thus, it would be necessary to either determine the most used traditional option or to compare the proposed solution with different groups of users that relies on different conventional methods.

Future works will be aimed at improving the proposed system depending on the feedback provided by the user in the assessment phase. Some Should requirements can be developed to improve the system further. Voice interaction is the only Could requirement not yet available in the system. Gesture interaction is the only Must requirement not yet implemented: replacing the VIVE controllers with finger-tracking gloves such as the Manus Metagloves[4] may enhance the easiness of use further. Tracking the user's waist and feet using three VIVE trackers could provide a complete embodiment system and let the user define the character pose simply by enacting it. Finally, investigating a method to intuitively define the characters' facial expressions might benefit the system.

[4]https://www.manus-meta.com/

REFERENCES

[1] Berford, B., Diaz-Padron, C., Kaleas, T., Oz, I. and Penney, D., 2017. Building an animation pipeline for vr stories. In ACM SIGGRAPH 2017 Talks (pp. 1-2).
[2] Henrikson, R., Araujo, B., Chevalier, F., Singh, K. and Balakrishnan, R., 2016, October. Multi-device storyboards for cinematic narratives in VR. In Proceedings of the 29th Annual Symposium on User Interface Software and Technology (pp. 787-796).
[3] Vogel, D., Lubos, P. and Steinicke, F., 2018, March. Animationvr-interactive controller-based animating in virtual reality. In 2018 IEEE 1st Workshop on Animation in Virtual and Augmented Environments (ANIVAE) (pp. 1-6). IEEE.
[4] Galvane, Q., Lin, I.S., Argelaguet, F., Li, T.Y. and Christie, M., 2019, March. Vr as a content creation tool for movie previsualisation. In 2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR) (pp. 303-311). IEEE.
[5] TiltBrush. http://www.tiltbrush.com/.
[6] Oculus. https://www.oculus.com/story-studio/.
[7] FrameForge, https://www.frameforge.com/.
[8] ShotPro, https://www.shotprofessional.com/.
[9] Marco Scarzello, Advanced Storyboard: Automatic storyboard generation using direct character control, http://webthesis.biblio.polito.it/id/eprint/25433
[10] Clegg, Dai; Barker, Richard (1994). Case Method Fast-Track: A RAD Approach. Addison-Wesley. ISBN 978-0-201-62432-8.
[11] Bittner, Kurt; Spence, Ian (2002-08-30). Use Case Modeling. Addison-Wesley Professional. ISBN 978-0-201-70913-1.