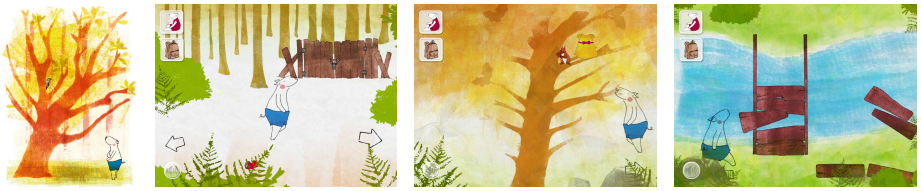
Real-time Hand-Painted Graphics for Mobile Games

Fabian Di Fiore[1], Tom Schaessens[1], Robin Marx[2], Frank Van Reeth[1], and Eddy Flerackers[1]

[1] Hasselt University - tUL - iMinds
Expertise Centre for Digital Media
Wetenschapspark 2
BE-3590 Diepenbeek, Belgium
{fabian.difiore, frank.vanreeth, eddy.flerackers}@uhasselt.be
http://www.edm.uhasselt.be

[2] LuGus Studios
C-Mine (Crib) 12
BE-3600 Genk, Belgium
robin@lugus-studios.be
http://www.lugus-studios.be



(a) (b) (c) (d)

Fig. 1. a) Actual scene from the Bo books. b–d) Snapshots from our mobile game.

Abstract. In this paper we set out to find a digital painting technique which allows the recreation of scenes from childrens books or graphic novels for usage in mobile games. The idea is to give the impression that scenes are being hand-painted at the moment the player is moving through them.
To this end, we propose a technique for digital painting, *mesh-based strokes*, that combines the strengths of existing painting techniques while keeping the limitations of mobile devices in mind. In this new technique, instead of representing strokes as pixels in a raster, strokes are created as flat meshes following the contour of a brush stroke. Furthermore, in close cooperation with artists digital painting and animation tools are introduced to facilitate the creation of new digital brushes and scenes in order to capture the style from children's books. In this paper, the Bo childrens book series is used as a case study [1].
We believe our system is effective in terms of ease-of-use, performs better on mobile hardware than traditional techniques and offers a new fresh perspective on stylised animation in mobile games.

Keywords: Computer Animation, Painterly Animation, Mobile Games

1 Introduction

Motivation. There are many picture books and graphic novels with very dynamic and interesting styles. When digitising these books, for animations or games, backgrounds and scenes are often represented as images that appear instantaneously or fade-in in some way. The entertainment industry, however, is always in demand for alternatives to the often used collage style of these games.

Scenes themselves, for instance, could be employed to support the progression of the story: as the player continues, more and more is revealed. Therefore, we will investigate the interesting possibilities that come to mind if scenes could be recreated in the same way the artist made them (e.g., line by line, stroke by stroke). In addition, even though mobile devices are becoming faster and more powerful, there is still a big performance difference compared to PCs. This is mainly due to the fact that mobile devices often have to operate on battery power, which means that low power usage is a priority which in turn comes at the cost of performance. Hence, we will need to pinpoint the main differences between mobile and PC architectures, and use this knowledge to optimise our implementation.

It is also our objective to allow the user to interactively create visually pleasing animations while keeping him/her in full control of the animation process. This is typically the case for smaller-scale productions where animators have to find their way more independently.

Contribution. In this paper we present a system to simulate hand-painted graphics in real-time. Our method allows for animated recreation of scenes and provides sufficient performance to be run on mobile devices. Along with the method of simulation, tools are developed to allow artists to quickly and intuitively create and animate scenes.

These tools will be integrated in the Unity3D game engine [2] which will be used in the creation of the Bo game.
Primarily our system features following characteristics:

- allowing traditional hand-painted strokes to be generated digitally;
- capturing the way an artist creates his picture books;
- animated recreation of captured scenes;
- optimisation for mobile devices;
- integration in existing engine (Unity3D)

The pictures in the inset (Figure 1) show an actual scenes from the Bo books as well as snapshots from the game created using our technique.

Approach. As a case-study the visual style of Bo will be recreated [1]. Bo is a childrens book series featuring Bo, a transparent piglet, as the main character (see Figure 1(a)). The books are aimed at children between the ages of 4 and 6, and teach them how to deal with emotions, such as love, fear and anger. Throughout the books multiple traditional painting techniques are combined; Bo

is drawn with a marker, backgrounds are painted abstractly with a paint roller, foreground elements are created using stencils and animals are often detailed watercolour paintings. This multitude of techniques makes Bo an ideal case-study.

In Section 2 existing techniques for digital painting will be analysed, and their feasibility within a mobile game determined. We will also look at common mobile hardware and its limitations. In Section 3 we propose a new digital painting technique, *mesh-based strokes*, based on the strengths and weaknesses of existing techniques. We will also elaborate on the digital copies of the brushes used in the Bo books and create the tools needed to let artists use these digital brushes. Next, we discuss how to animate the creation of strokes, and integrate this in a tool. In Section 4 we look at the results of our implementation. We compare the visual results, benchmark scenes created with the tool and analyse the usability of the tools themselves. Finally, Section 5 is our concluding section in which we also set the context for future work.

2 Related Work

In this section we discuss different methods for creating digital strokes. We will analyse the strengths and weaknesses in the light of real-time game performance and look at the software and hardware that will be used in the implementation of our proposed technique, to get a clearer image of pitfalls that might be faced.

Hand-painted Look. A hand-painted look typically is achieved through *physical simulation*, *non-photorealistic rendering (NPR)* or *raster graphics*.

Regarding physically simulating the inner workings of paint, a great advantage of these techniques is that often the results are barely distinguishable from reality [3,4,5]. Also, since the techniques are simulating all the paint interactions from the brush to the canvas as it would happen in real life, they lend themselves well to create an animated recreation of a scene. This physical accuracy, however, comes at a cost. Most of these techniques can not be considered real-time, at least not from the standpoint of a game developer.

Non-photorealistic rendering techniques [6,7,8] make use of (3D) geometric models and, hence, can use the graphics hardware to do the heavy lifting allowing to run real-time, even on modern mobile devices. A downside to rendering geometry is that, even though the effect of paint strokes can appear in the final image, there is no trivial way to recreate an object stroke by stroke. This makes an animated recreation much more difficult.

Images can also be created and edited using raster-based image editors [9,10,11]. These rely heavily on writing and reading pixels and support multiple layers that can blend with one another using various blending modes. A clear benefit of raster graphics techniques is the real-time visualisation of what the artist is painting, in the same way as it will appear in the final image. However, they do lack the accuracy of physical simulation as there is no flow of medium, or movement of pigment particles.

Mobile GPU Performance. Since our system will be implemented on mobile devices, more specifically the iPad2, its crucial to get a clear understanding of the capabilities and limitations of the hardware. All of the iPad devices, as well as many other tablets and smartphones use the PowerVR graphics architecture [12].

The PowerVR architecture deviates from standard immediate mode rendering (IMR) graphics hardware in the way that it uses tile-based deferred rendering (TBDR) as a method of 3D rendering. The core design principle behind the TBDR architecture is to reduce the system memory bandwidth required by the GPU to a minimum, as transfer of data between system memory and GPU is one of the biggest causes of GPU power consumption. This makes TBDR-based architectures very interesting for mobile devices.

3 Approach

In this section we will focus mainly on recreating the visual style of Bo, as a case-study, even though the results of our technique can be used to simulate many different painting techniques.

The goal of recreating the visual style of Bo is not the physically accurate recreation of scenes rather than recreating the global appearance of the books, and doing so as if the scene was drawn live.

3.1 Mesh-based Strokes

Following our discussion on achieving a hand-painted look (see Section 2), raster based graphics seems to be the best option as it allows for animated recreation of a scene, elements within a scene can change position — given that they are on a separate layer — and the artist is given a lot of freedom and control in the final result.

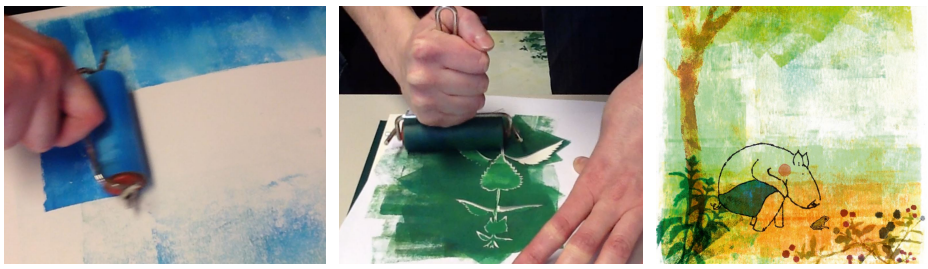
Benchmarking tests indicated that the combination of the ARM and PowerVR architectures is not very efficient at write-to-texture operations. Therefore, to combine the best of both worlds, we introduce *mesh-based strokes* which is a combination of NPR and raster graphics. In this new technique, instead of representing strokes as pixels in a raster, strokes are created as flat meshes following the contour of a brush stroke. This allows the same stroke-by-stroke creation as raster graphics or physical simulation. Creating the colour of the strokes is done through a GPU shader program, similar to the techniques used in NPR.

This allows us to exchange the slow write-to-texture operations with fast rendering of meshes on the GPU, while still allowing a stroke-by-stroke recreation. So we maintain the performance benefits of both NPR, and the stroke-by-stroke animatability of raster graphics.

3.2 Creating a Hand-painted Look

Introduction to the Visual Style of Bo. The Bo childrens book series has a distinct graphical style. We identified several main characteristics to this style.

A *rubber paint roller* is used mainly for background elements such as the sky, the ground and foliage, setting the global atmosphere for each image (Figure 2(a)). To add more details to the scene, *paper cut stencils* are used, and brushed over using the paint roller (Figure 2(b)). All animals and insects in the Bo books are drawn realistically. They are painted separately, scanned and added into the final image using image processing software. Most other elements in the scenes are transparent (e.g., trees show through foliage and foreground objects blend with the background). In the final composition, the artist plays around with different layer blending modes, until the wanted result is achieved (see Figure 2(c)).



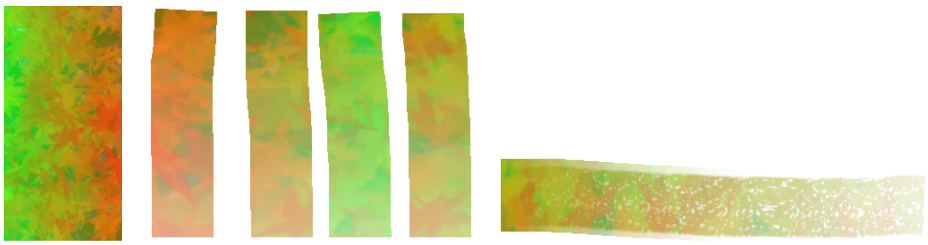
(a) (b) (c)

Fig. 2. Visual style of Bo. a) Rubber paint roller. b) Rolling the roller over a stencil. c) Final composition showing the blending of transparent elements.

Creating a Base Stroke. First, when drawing a stroke, a list of input points is sampled together with pressure sensitive (used to determine the width) and tilting data. The boundaries of these so-called brush footprints are then used to create the geometric mesh. Note that we do not use all the input samples directly as it would create a mesh with too many subdivisions. To prevent this, a minimum threshold distance is set between adjacent points.

One of the most clear characteristics of brushes is the fact that they contain a limited supply of paint. Hence, over time a brush stroke will fade out. Conceptually it seems easy to create a stroke that fades out over the length of a mesh by simply increasing the transparency gradually. However, as a shader has no context, it does not know if its drawing at the start or end of a stroke. Our solution lies in the usage of *uv*-coordinates: by adding a length-parameter as an additional *uv*-coordinate, fading out the stroke becomes possible.

The resulting mesh contains enough information to allow a shader to simulate common characteristics of brushes. The mesh itself allows to move back and forth, as one would with a brush or a paint roller, and the length-parameter gives the shader information about how far along a mesh it is painting allowing strokes to fade out.



(a) (b) (c)

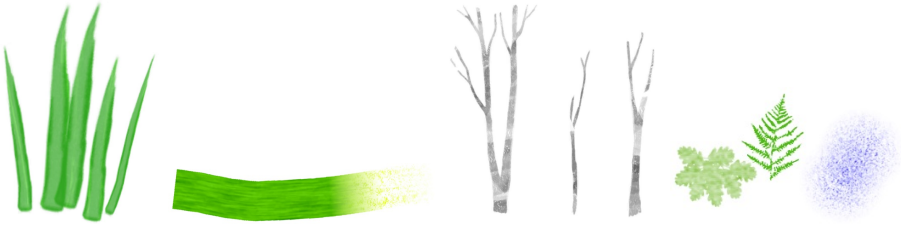
Fig. 3. Rubber Paint Roller. a) Image of a wide paint roller. b) Multiple differently coloured strokes using the same paint roller. c) Image showing the dried pattern within a brush stroke.

Rubber Paint Roller. The fading of the roller strokes is implemented as a linear interpolation of the alpha value, according to the length-parameter set in the stroke mesh. In order to give the artist more freedom, two more parameters are introduced. *Fade offset* allows the stroke to be more transparent when beginning to draw whereas *fade multiplier* allows control over the speed at which a stroke fades out. Using these two parameters, a transparent, slowly fading stroke can be created that is very suitable for backgrounds, where individual strokes must not stand out. A thick, more opaque roller can be created for creating more prominent features of a scene.

As each stroke of the roller is unique, paint should never appear on the canvas in exactly the same way. Due to the way paint is applied to the physical roller, namely two or more dollops of paint next to each other through which the roller is rolled, there are three general blends of paint: (i) mostly one colour, with traces of the other, (ii) an even blend of both, or (iii) mostly the other colour with traces of the first. Figure 3 illustrates this by means of a very wide paint roller. When rolled through two dollops of paint, each side of the roller will have the colour of the paint that was most to its side, while the centre will have a patchy blend of both colours (Figure 3(a)). To create unique strokes each time using the same roller, for each stroke only a single narrow band is selected from the wide roller (Figure 3(b)).

Another feature is that when rolling back and forth in a fanning pattern, a pattern of white dots is created. This is due to the relief created by the paint itself as thin layers of dried paint on the surface of the roller prevent paint in between from touching the paper. To simulate this effect, we employ a black and white mask: white representing spots where the paint should adhere, black where it doesn't. This mask is added as a texture to the shader and multiplied with the transparency value (Figure 3(c)).

Other Brushes. Since the mesh which serves as a base for the brush strokes was created to be reusable, some other brushes were created as well. Here again, the brushes are based on brushes often found in the Bo books.



(a) (b) (c) (d)

Fig. 4. Different kinds of brushes. a) Grass brush. b) Oily brush. c) Example of a stroke using the stencil roller brush with a birch tree stencil. d) Stamp brushes with various different masks.

The *grass brush* creates the effect of individual blades of grass (Figure 4(a)). It is achieved by ripping of a corner of paper, dipping this in paint and quickly dragging it over the canvas. To simulate the effect of this brush, a mask was applied and stretched over the entire length of a stroke based on the length-parameter from the mesh. This mask has a transparent inside and a defined outside similar to that of the real strokes. By stretching the mask over the length of the stroke, different sizes of grass can be created with the same shader.

The *oily brush* is a brush that was used in one of the Bo books to create the effect of thick grass. The brush clearly has visible brush fibres in the strokes and starts packed with thick paint. As pressure on the brush lessens, the brush becomes more transparent and grainy, until only the longest fibres are touching the paper. The stroke ends in only a few fibres still touching (Figure 4(b)). Two adjustable parameters were introduced: the (i) percentage to graininess to determine at which percentage of the stroke the thick paint ends and the graininess of the stroke becomes clear, and (ii) the percentage to fade to determine at which remaining percentage the stroke fades out to just a few brush fibres.

The *stencil roller* is a variation on the original paint roller to create the effect of rolling over a stencil (Figure 4(c)). The stencil is applied to the canvas and held in place while the paint roller is applied. The inside of the stencil has the same characteristics as the paint roller. The outside of the stencil has a double border: one as the hight of the stencils paper prevents the roller from reaching the canvas and another where the fibres on the edge of the stencil absorbed paint. To achieve this effect, a world-space mask was created, stating where paint from the roller could be seen, and where it couldn't. An extra texture in the paint roller shader serves as this mask. To achieve the same effect as a real stencil, the transparency value of the paint roller is changed according to the value of the stencil. Whenever the stencil is black, the alpha is kept as it is, whenever the stencil is white, the alpha is set to 0.

Stamps are the only types of brushes that do not use the same mesh as the paint roller, since stamps are not drawn as strokes but placed at once. Because of this, stamps are created by simply generating a quad mesh. To this mesh, a

shader is added with a black and white mask; this masks shows the colour of the stamp where the colour is white, and is transparent where the colour is black. This allows any shape of stamp to be made, ranging form a sponge-like pattern to detailed leaves (Figure 4(d)).

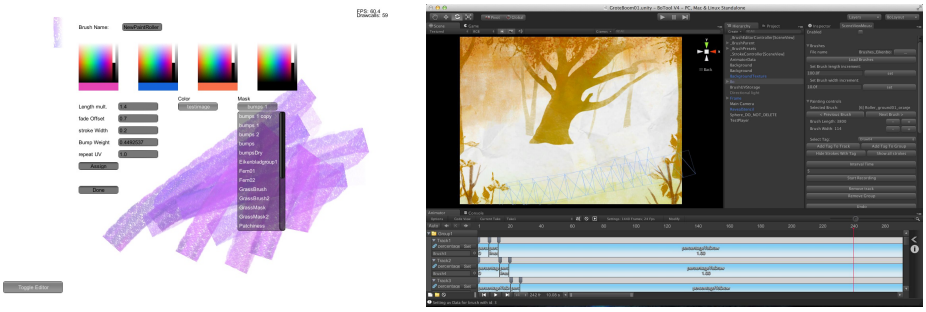
3.3 Content Creation Pipeline Toolset

In order to allow the artist to create scenes, a toolset consisting of 2 separate tools was made: the brush editor and the animation tool.

Brush Editor. The brush editor allows artists to manage brushes. Brushes in this case are simply the collection of a shader and all of their parameters and textures.

For each scene the artist is restricted to one set of brushes. This is to ensure that the amount of different shaders used in a scene remains low, which optimises the amount of batched draw calls. A set of brushes can be used for multiple scenes. In the brush editor the artist can create new sets of brushes, as well as update previously created sets. A preview canvas is provided to test out new brushes, and to see how different brushes match together. There is also a simplified version of the painting controls from the animation tool, allowing the artist to set the length and width of strokes in order to preview those modifications accurately.

A simple editing screen is provided to change settings for the brush. These settings are a more user-friendly representation of the parameters used in the shaders. Here the artist can also select which textures to use. A screenshot of the Editor tool can be seen in Figure 5(a).



(a) (b)

Fig. 5. a) Screenshot of the Bo brush editor tool. Here the settings for a roller brush are edited. b) Screenshot of the Bo animation tool.

Animation Tool. The animation tool allows the artist to draw scenes and animate the way in which they appear (Figure 5(b)).

The first step when beginning to create a scene is to load a set of brushes. Often brushes are themed to a certain setting, e.g., an autumn forest or a birch tree forest. Once brushes are loaded, the artist can select which brush to use, set the length and width for a stroke, and start painting. While painting, positions of points within the strokes are saved. This allows to recreate a stroke exactly as it was drawn. Besides this, the settings at the moment of drawing are saved. These settings include the selected brush, as well as the width and length for the stroke.

Another part of the tool is the animation timeline for which we used a free open-source timeline for Unity3D [13]. Strokes are given the ability to be added to the timeline. When added, a variable *percentageToDraw*, ranging from zero to one, can be modified through key frames. Each time the percentage to draw variable is altered, a new mesh, of which the length is that percentage of its total length, is generated. This allows the artist to make the stroke appear when and how he wants.

Recording functionality is also integrated in the tool. By default, each new stroke starts one frame after the previous one has ended. Afterwards the artist can fine-tune the effect by moving around key frames in the timeline.

4 Results

In this section we look at some of the scenes created using our mesh-based strokes technique. Figure 6 shows some actual scenes from the Bo books while in Figure 7 final snapshots of the Bo game can be seen.

On average, each scene contains 4500 vertices and 2400 triangles. Initially, running on an iPad2, the performance was quite low: the frame rate dropped quickly below 30 fps ending up at around 5 fps. This drop in frame rate is mainly due to overdraw: each stroke covers a significant amount of the screen and there is much overlap of strokes.

As a result, we examined the influence of render-to-texture techniques to speed up the drawing of strokes. Render-to-texture works by taking the pixel seen through the viewport of a camera and rendering this to an image instead of rendering it to the screen. We use render-to-texture to flatten all background strokes by rendering them all to one texture and then hiding the original strokes. This ensures that instead of numerous overlapping transparent strokes, only one opaque background has to be rendered. This greatly improves performance yielding a consistent frame rate of 60 fps, even when animating.

Also, due to delegating most of the heavy lifting to the GPU, the CPU is still available to handle the game logic.

5 Conclusion and Future Work

In this paper we presented a digital painting technique which allows the recreation of scenes from childrens books or graphic novels for usage in mobile games.

Our technique allows a very flexible animation of strokes. This combined with our editing and animation tools gives the artist complete control over the animation of a scene. Through continuous optimisation of our techniques, we manage to stay well above the lowest acceptable frame rate of 30 fps making it run smoothly on mobile devices.

Comparing our results to the original images of the case study, we can conclude that our brushes and results resemble those of the original closely.

5.1 Discussion

One of the main issues with the tool at this time is the inability to use Unity3Ds undo functionality. At this point, it is not possible to detect an undo event consistently within Unity3D as the event is only sent to windows that have focus at the time of undoing. Since the drawing canvas is separate from the rest of the tool, no undo event is registered when drawing. This problem is partially omitted by providing custom undo and redo buttons with the required additional functionality.

Another disadvantage is that the artists have less creative freedom than when using off-the-shelf digital tools. Our tool, however, provides enough functionality to create scenes without any additional bells and whistles.

5.2 Future Work

A first possible improvement is broadening the amount of different brushes that can be simulated. There are many more styles to explore that can be recreated using the techniques from this paper including a multitude of other childrens books, as well as many graphic novels.

Improvements to the dynamic creation of the mesh can also be made. The current version works by generating a subdivision at certain distance intervals, this could be changed to a technique that bases subdivisions on the amount of curvature of a stroke. This would drastically lower the amount of geometry when drawing straight lines and create much smoother curves.

There is also still room for improvement in the brush creation and animation tools such as a movieclip system allowing the reuse of small animations.

Acknowledgements. We gratefully express our gratitude to the European Fund for Regional Development (ERDF) and the Flemish Government, which are kindly funding part of the research at the Expertise Centre for Digital Media.

References

1. Varkentje Bo. World Wide Web, `http://www.varkentjebo.net/`, 2014.
2. Unity3D. World Wide Web, `http://www.unity3d.com/`, 2014.
3. Cassidy J. Curtis, Sean E. Anderson, Joshua E. Seims, Kurt W. Fleischer, and David H. Salesin. Computer-generated watercolor. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '97, pages 421–430, New York, NY, USA, 1997. ACM Press/Addison-Wesley Publishing Co.
4. William V. Baxter, Jeremy Wendt, and Ming C. Lin. IMPaSTo: A realistic model for paint. In *Proc. of Symposium on Non-Photorealistic Animation and Rendering (NPAR)*, pages 45–56, June 2004.
5. Tom Van Laerhoven and Frank Van Reeth. Real-time simulation of watery paint. *Computer Animation and Virtual Worlds*, pages 429–439, 2005.
6. Johan Claes, Fabian Di Fiore, Gert Vansichem, and Frank Van Reeth. Fast 3D cartoon rendering with improved quality by exploiting graphics hardware. In *Proceedings of Image and Vision Computing New Zealand (IVCNZ2001)*, pages 13–18. IVCNZ, November 2001.
7. Bruce Gooch and Amy Ashurst Gooch. *Non-Photorealistic Rendering*. A. K. Peters Ltd., ISBN: 1568811330, 2001.
8. Su Ian Eugene Lei and Chun-Fa Chang. Real-time rendering of watercolor effects for virtual environments. In *Proceedings of the 5th Pacific Rim Conference on Advances in Multimedia Information Processing - Volume Part III*, PCM'04, pages 474–481, Berlin, Heidelberg, 2004. Springer-Verlag.
9. GIMP. World Wide Web, `http://www.gimp.org/`, 2014.
10. Corel Painter. World Wide Web, `http://www.corel.com/corel/`, 2014.
11. Adobe Photoshop. World Wide Web, `http://www.photoshop.com/`, 2014.
12. Imagination Technologies. Powervr series5 graphics sgx architecture guide for developers, 2011.
13. Adobe Photoshop. World Wide Web, `http://forum.unity3d.com/threads/135982-Animator-The-Ultimate-Timeline-Cutscene-Editor-for-Unity`, 2014.

Fig. 6. Some actual scenes from the Bo books.

Fig. 7. Snapshots from the game created using our technique.