Dynamic Canvas for Non-Photorealistic Walkthroughs

Matthieu Cunzi †⊙ Joëlle Thollot † Sylvain Paris † Gilles Debunne †
Jean-Dominique Gascuel † Frédo Durand ⋄
† ARTIS/GRAVIR ⋄ MIT ⊙ REVES/INRIA Sophia-Antipolis

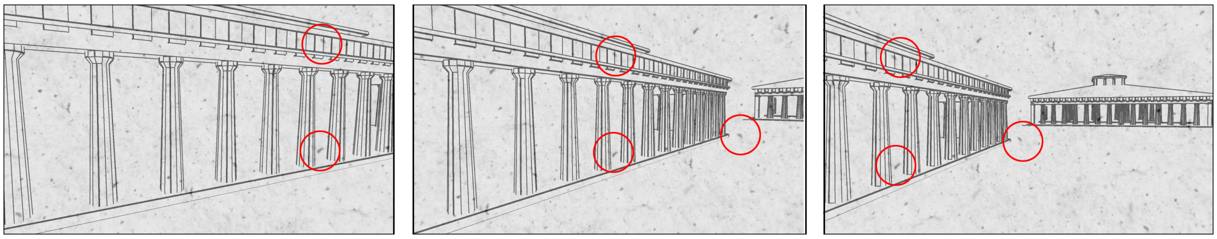


Figure 1: Walkthrough with a dynamic canvas. Note how the grain of the paper follows the strokes.

Abstract

The static background paper or canvas texture usually used for non-photorealistic animation greatly impedes the sensation of motion and results in a disturbing "shower door" effect. We present a method to animate the background canvas for non-photorealistic rendering animations and walkthroughs, which greatly improves the sensation of motion and 3D "immersion". The complex motion field induced by the 3D displacement is matched using purely 2D transformations. The motion field of forward translations is approximated using a 2D zoom in the texture, and camera rotation is approximated using 2D translation and rotation. A rolling-ball metaphor is introduced to match the instantaneous 3D motion with a 2D transformation. An infinite zoom in the texture is made possible by using a paper model based on multifrequency solid turbulence. Our results indicate a dramatic improvement over a static background.

1 Introduction

The field of *Non-Photorealistic Rendering* [GG01, SS02, Rey02, Dur02] not only captures the qualities of traditional media, but also permits their animation. Media that were inherently static come to life and can be animated and used for interactive walkthrough. This raises a number of challenges: How should the elements of the picture such as strokes (marks) or background paper be animated, and how can we ensure *temporal coherence*? Two basic strategies are possible, and neither of them is perfect. One can either attach the marks to the 2D space of the picture, or attach them to the 3D objects. In the first case, the scene appears to be viewed through a shower door, and in the second, the 3D objects seem to be textured with, or carved in, artistic strokes. The problem boils down to the tension caused by the dualism of pictures, both 2D compositions and representations of 3D scenes [Dur02].

In previous work, much attention has been paid to strokes and their temporal coherence, but the temporal behavior of the background canvas or paper has been mostly ignored. As a result, most NPR animations or walkthroughs seem to be projected on a paper or canvas texture using a slide projector, and the background does not participate in the animation or walkthrough experience. The strokes slide on the paper, which not only reduces the "immersion" and motion cues, but also impedes the sense of the picture as a whole, because paper and strokes do not interact and seem to lie in two different dimensions.

In this paper, we present a *dynamic canvas* where the background texture is animated to provide strong motion cues and bind the background and moving strokes. It dramatically improves the "immersive" impression and motion cues for non-photorealistic walkthroughs, and dramatically reduces the effect of strokes that slide on the background. Our method presents a careful balance between the 2D qualities of the background texture and the 3D motion of the observer. This is achieved by panning around and zooming in a 2D background paper texture in order to approximate 3D motion. The problem can be stated as follows: The motion of the observer is three-dimensional and results in a complex optical flow, including parallax effects. In contrast, the canvas or paper of a picture is characterized by its flat and rigid 2D quality. Our goal is to use a rigid 2D motion for the paper in picture space that provides a perceptual impression of motion as close as possible to the 3D displacement in object space.

1.1 Related work

The issues of animation and temporal coherence have received much attention in non-photorealistic rendering. We review the techniques most relevant to our approach, and we refer the interested reader to the books covering the field [GG01, SS02].

Meier [Mei96] rendered animations of 3D models in a painterly style using particles. Related techniques were used in the DeepCanvas system at Disney, where the brush strokes drawn by animators were attached to a 3D model [Dan99]. Curtis also uses particles to create loose and sketchy animations [Cur98]. Some techniques process video sequences into a painterly style. Temporal coherence can be ensured by using, e.g., optical flow [Lit97] or image differences [HP00].

For interactive NPR applications, temporal coherence and constant screen-space size of marks can be obtained by an appropriate use of the hardware mipmapping capabilities, as proposed by Klein et al. [KKL+00] and later refined by Praun et al. for pen-and-ink styles [PHMF01, WPFH02]. To ensure that the screen-space size of their mark remains relatively constant, they double the size of the marks for each mipmap level. Our solution for forward translation is related to their technique, in that our paper texture is self-similar and ensures a constant screen size of the grain.

The interaction between the paper and the marks has been carefully simulated or imitated, e.g., [LMHB00, MG02, SB00, CAS+97], but to the best of our knowledge, no previous work has addressed paper motion for NPR walkthroughs with arbitrary camera motion.

The technique most related to our approach is the work on multiperspective panoramas by Wood et al. [WFH+97]. Inspired by traditional cel animation, they build a single 2D background to simulate a complex 3D path. The effect is obtained by sliding a rectangular window on a multiperspective panorama. The non-linear projection is computed by matching the optical flow of the simulated 3D path using a least-squares approach. In this paper, we also propose to approximate complex 3D movements with 2D transformations on a background texture, but in an interactive context for arbitrary paths.

1.2 Method overview

Our goal is to preserve the 2D quality of the background texture while providing compelling 3D motion cues and avoiding the effect of strokes that seem to "slide" on the paper. The exact definition of "2D quality" will be described shortly and is one of the key decisions that influences the motion depiction style. The camera motion usually results in a complex optical flow that does not correspond to a simple 2D transformation. Matching the 2D paper motion with the complex movement of the 3D observer or with object motion is therefore usually overconstrained. The grain of the paper cannot follow the 3D points on the scene objects without distorting the paper texture. Therefore, a compromise has to be chosen, and the concessions will depend on the task and style. We propose a solution that is a good tradeoff for most situations, but the infinite range of possible strategies is an example of the richness of non-photorealistic styles.

Our method can be intuitively expressed for the two most basic types of observer motion: camera rotation and forward translation. When the camera is translating forward, we exploit the ambiguity between 2D zoom and forward translation. We zoom in the background texture, which provides a strong forward motion cue. In fact, this approximates the visual impression of moving in a snow storm or in the fog. Note that in contrast to Wood et al. [WFH+97], we use the zoom not only to simulate a change in the focal length, but also to simulate forward translation. In order to implement our zooming approximation of translation, we need to be able to infinitely zoom in the background texture. This will be described in Section 5.

When the camera rotates, the projective picture (i.e. the picture projected on the sphere of directions) should not be altered since the eye position is fixed. In particular the strokes used to draw the 3D model should not seem to slide on the background paper. The screen motion of the paper should match the screen motion of the 3D model induced by the camera rotation as much as possible. We therefore want to rotate the background texture as if it were projected on a sphere centered on the viewpoint. It is well known that, unfortunately, texture mapping on a sphere raises singularity problems at the poles. Moreover, combining the 2D zoom for translation with the spherical rotation is far from trivial. Indeed, the zoom is inherently a linear transformation of the Cartesian plane. In contrast, the rotation is defined on the sphere of directions, which produces non-linear transformations in the plane. Both the topology (infinite plane vs. sphere) and the linearity issue make the compatibility between the two approaches difficult. This is why we developed an approximation strategy to map the entire technique to the Cartesian 2D plane.

Following Wood et al. [WFH+97], we introduce a general framework for the study of temporal coherence in NPR, using the notion of *motion field*. This allows us to formally justify our method and to numerically link the 2D transformation of the background and the 3D motion.

This paper makes the following contributions:

- We show how simple 2D transformations of the background paper can dramatically enhance the motion cues for interactive NPR walkthroughs.

- We introduce the *rolling-ball* metaphor to match the spherical trajectory of the center of the screen to a Cartesian 2D displacement.
- We warp the result of the above motion to better match the motion cues for camera rotations.
- We introduce a technique to perform an infinite zoom into a 2D texture defined procedurally or using a scanned image.

The paper is organized as follows: In Section 2, we introduce the framework of motion fields. In Section 3, we propose a simple technique to approximate the motion field of a 3D displacement with 2D similarity transforms. In Section 4 we introduce a screen-space spherical warp in order to better match the 3D motion cues for rotations. Section 5 presents our self-similar paper model that permits infinite zooms. We describe our implementation and results in Section 6 before concluding.

2 Motion field and 3D motion cues

The *motion field* [Hor86] represents the time derivative of the 2D projection of a 3D point. The motion field is slightly different from the optical flow, which is derived from image brightness, while the motion field is purely geometric. As shown by the landmark work by Gibson, [Gib79] the motion field and optical flow are crucial visual cues for moving observers. The qualitative pattern of the motion field is a strong indication of the motion of the observer (*egomotion*), or of the motion of dynamic objects in the scene. In this paper, we focus on the motion of the observer, but this framework can be used to study temporal coherence with any motion.

To express the motion field equations [Hor86], we consider a point $M = (X,Y,Z)$ in a camera coordinate system that projects on a screen at distance f on a point m with coordinates (x,y,f).

We first compute the motion field for a translation along the Z axis at speed $\frac{dZ}{dt}$. From the classical relations $x = \frac{fX}{Z}$ and $y = \frac{fY}{Z}$, we get:

$$\begin{aligned} \frac{dx}{dt} &= -\frac{fX}{Z^2}\frac{dZ}{dt} = -\frac{x}{Z}\frac{dZ}{dt} \\ \frac{dy}{dt} &= -\frac{fY}{Z^2}\frac{dZ}{dt} = -\frac{y}{Z}\frac{dZ}{dt} \end{aligned} \tag{1}$$

The motion field for the forward translation towards a vertical plane parallel to the image plane is depicted in Fig. 3(a). It is a radial field. In contrast, the motion field for the translation towards a more complex scene, such as a sphere and a plane (Fig. 3(c)) exhibits a more involved pattern and discontinuities along silhouettes. Such motion fields typically cannot be matched with simple 2D transformations.

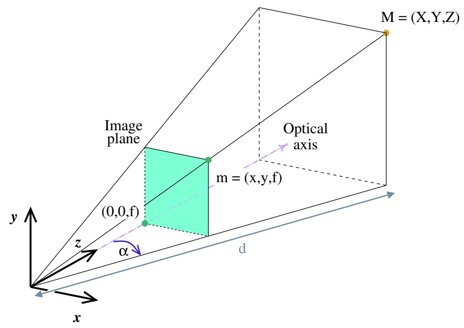


Figure 2: Motion field calculation for a pinhole camera.

We also compute the motion field for a rotation around the Y axis at speed $\frac{d\alpha}{dt}$ (see Fig. 2). We introduce the 3D distance $d = \sqrt{X^2+Z^2}$ and obtain the relation $Z = d\cos\alpha$. We can express both x and y as function of α:

$$x = f\tan\alpha \qquad\qquad y = \frac{fY}{Z} = \frac{fY}{d\cos\alpha}$$

and obtain the final result:

$$\frac{dx}{dt} = \left(f + \frac{x^2}{f}\right)\frac{d\alpha}{dt} \qquad\qquad \frac{dy}{dt} = \frac{xy}{f}\frac{d\alpha}{dt}$$

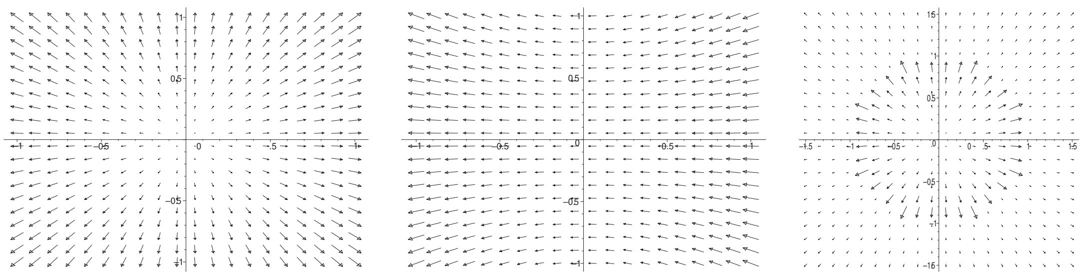
Figure 3(b) shows a motion field for a rotation.

3 Matching the motion field with 2D similarity transforms

We want to match or approximate such motion fields by the motion field induced by simpler 2D transformations. This will permit the animation of the background paper or canvas in a 2D fashion, while providing the user with convincing 3D motion cues. We need to define what "2D quality" or "simple 2D transformation" mean, and therefore which constraints must be enforced on the motion. In contrast to Wood et al. [WFH+97], our approximation is computed directly from the motion and is not a least-square optimization.

The most immediate choice is to allow only 2D *similarity* transformations [WFH+97]. Similarity transforms are composed of a rigid transformation and an isotropic scale. They completely respect the rigid 2D nature of the background. We first discuss the simple cases of translation along the optical axis and rotations around the vertical axis, before discussing the general case.

The motion field of a 3D forward translation towards a vertical plane can be perfectly matched by a zoom in the background, that is, by a 2D scaling, as suggested by Fig. 3(a). However, we cannot match a complex motion field such as the one shown in Fig. 3(c) with a simple 2D transform. Our technique chooses a *subjective distance*,



(a) (b) (c)

Figure 3: (a) Motion field for a forward motion towards a vertical plane. (b) Motion field for a camera rotation. (c) Motion field for a forward translation towards a sphere in front of a plane. Note the discontinuity and parallax effect at the silhouette of the sphere.

D, for which the motion field is correctly matched. For example, in Fig. 3(c), we can choose to match the motion field of the sphere or the motion field of the background plane. We propose two strategies to choose this subjective distance. It can be set constant, to approximate a "fog" at a given distance, or it can track a given object of interest.

The motion field for a camera rotation along the vertical axis (Fig. 3(b)) cannot be easily approximated, since it describes hyperbolae. However, if the field of view is not too wide, these hyperbolae are very flat and close to horizontal lines. The similarity that can best approximate such motion fields is a horizontal translation. In Section 4, we will introduce a spherical warp to compensate for this approximation.

In fact, the problem we are trying to solve for camera rotations can be simply visualized by considering the sphere of directions, that is, an arbitrary sphere centered on the observer. We can map the sphere of directions with a paper texture to perfectly respect the camera rotation. But we choose to approximate such a sphere rotation by a planar motion. In addition to alleviating the pole problem, it respects the 2D quality of the paper better, and it makes it possible to combine this transformation with our 2D zoom.

3.1 Matching the translation

We decompose the translation of the observer into two components: one along the optical axis, and one translation parallel to the plane of the image.

Translation along the optical axis

The forward translation along the optical axis is matched with a 2D zoom. In order to numerically relate them, we study the motion field (i.e. the screen velocity) of a 2D zoom. If the camera is translating towards a plane at distance D, from (1) we have:

$$\frac{dx}{dt} = -\frac{x}{D}\frac{dZ}{dt} \quad \text{and} \quad \frac{dy}{dt} = -\frac{y}{D}\frac{dZ}{dt} \tag{2}$$

To obtain the relation between two consecutive frames t and $t+\Delta t$, we consider a camera translating at constant speed $\frac{dZ}{dt} = \zeta$ during this short Δt. We simplify (2) and obtain $\frac{dx}{dt} = k\,x$ and $\frac{dy}{dt} = k\,y$ with $k = -\frac{\zeta}{D}$, which leads to the classical differential equation solution:

$$\begin{pmatrix} x(t+\Delta t) \\ y(t+\Delta t) \end{pmatrix} = e^{k\Delta t} \begin{pmatrix} x(t) \\ y(t) \end{pmatrix} \tag{3}$$

This shows that we can match the motion field exactly with a zoom at exponential rate k.

$\zeta\,\Delta t = \Delta Z$ is the signed distance corresponding to the forward translation between the two frames. We therefore have the final relation:

$$\begin{pmatrix} x(t+\Delta t) \\ y(t+\Delta t) \end{pmatrix} = e^{-\frac{\Delta Z}{D}} \begin{pmatrix} x(t) \\ y(t) \end{pmatrix} \tag{4}$$

In the above equations, the *subjective distance* D, plays a major role. The motion field of objects at distance D is correctly matched. It relates the zoom rate to the 3D scene, and a shorter subjective distance results in a faster motion cue. The choice of the subjective distance is an important decision. It is application- and style-dependent.

Translation parallel to the image plane

The translation component parallel to the image plane at speed $(\frac{dX}{dt}, \frac{dY}{dt})$ is matched with a 2D translation where the translation vector has been scaled using the perspective ratio at the subjective distance D:

$$\frac{dx}{dt} = -\frac{f}{D}\frac{dX}{dt} \qquad \frac{dy}{dt} = -\frac{f}{D}\frac{dY}{dt}$$

For a general translation, this results in off-center zooms because of the composition of the translation and the zoom. In this case, the final zoom center lies on the vanishing point of the translation direction, which is what we would expect intuitively.

3.2 The rolling-ball metaphor for camera rotation

For camera rotation, we decide to perfectly match the motion of the center of the image. We consider the Cartesian texture plane as tangent to the sphere of directions, with the tangent point at the center of the screen[1]. Our method locally approximates the sphere by this tangent plane (Fig. 4). The motion then corresponds to the sphere "rolling without sliding" on the plane, a classical situation in mechanics, e.g., [Fre65]. Matching the motion field for the center point between the sphere of direction and the texture plane means that the velocity on the sphere and on the plane are equal. This is the definition of rolling without sliding (a.k.a. without slipping). For example, we show in Fig. 4 that if the camera rotates around the vertical axis, the contact point on the texture plane is translated in the opposite direction. The trajectory of the center of the image in the texture plane is simply a straight line. This case is related to the cylindrical projection used in cartography to map the Earth, e.g., [Den96]. Note that after a camera rotation of 360 degrees, the background is usually not the same, unless the texture happens to be tileable with a period equal to the length of the equator of the rolling ball.

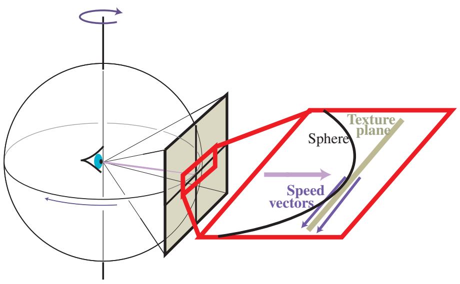


Figure 4: A ball rolling without sliding. The speed vectors of the sphere contact point and of the texture plane contact point are equal.

If the rotation axis is not orthogonal to the optical axis, the rolling motion describes a circle in the texture plane (Fig. 5a). The center of this circle is the intersection C of the rotation axis with the texture plane. The 2D transformation is the rotation around this intersection point. This corresponds to the conical projection in cartography [Den96]. Similar to the previous case, the background after a rotation of 360 degrees is usually not the same, because the length of the parallel circle of the sphere tangent to the texture plane is usually not the length of the circular trajectory in the texture plane (see Fig. 5a). This point will prove important in the next section.

As a special case, if the rotation axis is the optical axis (Fig. 5b), then the 2D texture plane undergoes the same rotation along the center of the image as the spherical rotation.

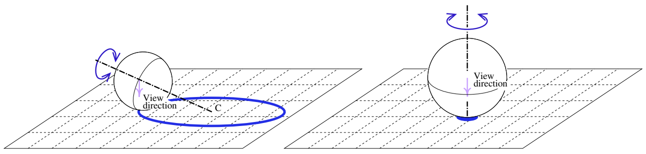


Figure 5: (a) A ball rolling on a plane with rotation axis intersecting the plane. (b) Extreme case where the optical and rotational axis are identical.

In the general case, we consider the instantaneous rotation between two adjacent frames. We use instantaneous Euler angles in the local frame of the observer. This does not suffer from the singularities of the Euler angles because we recenter the Euler axis for each frame, and because the rotation between two frames is usually less than 90 degrees. The details of the formulae for the 2D rigid transformation approximating camera rotation are given in appendix A.

4 Spherical warp

In the previous section, we have shown how zooming and the rolling-ball metaphor can approximate the motion field with simple 2D similarity transformations. The trajectory of the picture center is matched exactly. We now introduce a 2D warp that improves the approximation for the rest of the picture in the case of camera rotation. We focus on rotations that keep the horizon level. That is, the camera is allowed to rotate only along the vertical axis of world space (left-right rotation), or along the camera horizontal axis (up-down rotation). Our warp results in a perfect match for left-right camera rotation, and very good approximations for up-down rotation.

4.1 Basic warp

As discussed in Section 2, the motion field and the trajectories describe conics on the screen during a camera rotation (Fig. 3b). Moreover the speed varies along these conics. In our rolling-ball approximation, the trajectories are circles (with center the intersection of the rotation axis and the texture plane, see Fig. 5a) or lines, and they have constant speed. Our warp therefore maps circles or lines

[1] formally speaking, it is tangent to an embedding of the spheres of direction with radius f

in the 2D texture as obtained from Section 3.2 to the appropriate conics with varying speed.

Since we want to convey the impression that the paper texture is applied on a sphere centered on the camera, the warp is conceptually decomposed into two steps: the texture is mapped onto the sphere, and the sphere points are then mapped onto the screen (Fig 6). The trajectories on the sphere are parallel circles (latitude=cte on the Earth).

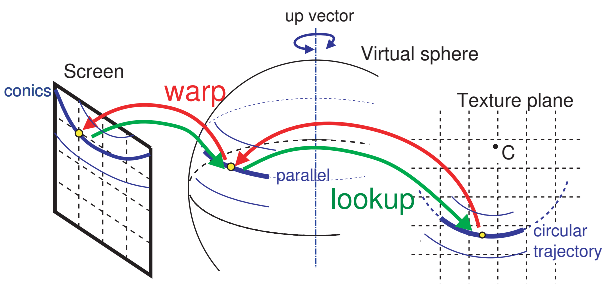


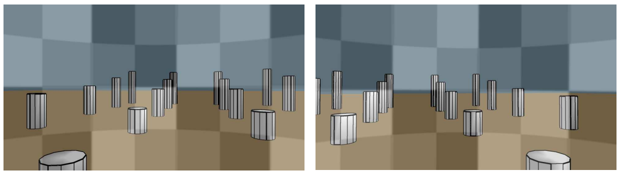
Figure 6: A warp in two steps: a screen point is mapped onto a point on a sphere parallel which is then locally mapped onto a circle in the texture plane.

In practice, we need to perform the inverse mapping operation (lookup), going from a screen point to a parallel on the sphere and finally to a circle on the texture plane. The first step is a simple spherical coordinate computation that goes from a screen point (x, y) to spherical coordinates (θ, φ), where $\varphi = cte$ describes the parallel and θ is the location on the parallel. Since we use the up vector as a reference, the center of the screen is at $(\varphi = \alpha, \theta = 0)$, where α is the tilt of the camera (elevation from the horizon).

For the second step, we map a parallel circle $\varphi = cte$ to a trajectory (circle or line) in the the texture (Fig. 6, right). Detailed formulae and illustrations are given in appendix B and Fig. 12. Note that for a parallel, we can choose any texture circle as long as the parallels are monotonously mapped to the circles. The meridians on the sphere defined by $\theta = cte$ corresponds in the texture to lines going through the intersection C of the rotation axis and the texture plane. We have one remaining degree of freedom: we need to decide which texture circle is mapped to which parallel.

We use this degree of freedom to optimize the motion field for up-down rotation. The spacing of the texture-space circle is chosen so that the motion field of the points of the vertical line in the center of the picture is matched perfectly. This also makes the pattern of vertical velocities for the whole picture qualitatively correct. That is, from top to bottom, the points slow down and then accelerate. The detailed formulae are given in appendix B.

The result of this warp is illustrated in Fig. 7 with a checkerboard texture for better visualization. The effect is best seen in the accompanying video published on the graphics interface web site.



(a) (b)

Figure 7: Between (a) and (b), the observer is looking at the right and thus the checkerboard moves to the left. Note the distortion of the texture such that no sliding occurs.

4.2 Case of closed trajectories

The above warp assumes that no screen trajectory is a closed curved (ellipse). Indeed, recall that when rotating by 360 degrees, the trajectory in the texture is likely not a full circle. This is not a problem with the basic rolling-ball technique because the texture is displayed “as is” after the similarity transform. But when combined to the warp, it can lead to discontinuities on the ellipse trajectories. This occurs when the vanishing point of the vertical direction (corresponding to C in the texture) is visible on the screen. For open trajectories, the problem does not occur, in a sense because the discontinuity happens off-screen.

We however note that no warp is required in the extreme case where C is in the center of the screen, that is, when the viewer is looking completely up or down. In this case, the rolling ball results in the exact motion field. Intuitively, when the observer is looking higher and higher, the screen trajectories transition from hyperbolae to ellipses and then to circles. Therefore, the circular trajectories described by the rolling ball become better and better approximations, and the warp is not required.

We therefore progressively turn the warp off when the observer is looking up. In practice, we simply use a $\sin^2$ interpolation that reaches zero when the vanishing point of the vertical reaches the screen.

5 Procedural zoomable paper

In order to produce our approximate motion field for forward translation, we need to be able to infinitely zoom in the paper texture. We want to ensure that for any zoom level, the texture looks like paper. This is related to self-similar fractals, e.g., [Hut81, Bar88], or to super-

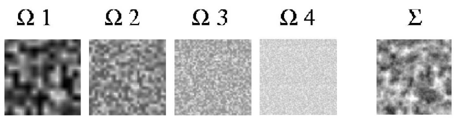


Figure 8: Successive octaves in a Perlin noise. On the right we show the summation of the four octaves.

resolution, e.g., [BK00]. We also need to be able to apply arbitrary rotations and translations. The latter can be obtained using tileable texture. We present a solution based on procedural solid textures [Per85]. We show that an infinite zoom can be obtained by cyclically shifting the frequency spectrum of a Perlin turbulence, and that the method can be applied to scanned textures as well.

5.1 Static paper texture

Our basic technique simulates a simple paper grain using a Perlin turbulence. The principle of solid turbulence is to synthesize and render complex noise using a sum of signals with different frequencies or *octaves* [Per85] (Fig. 5.1). We define a *base frequency* F_1 and a *base amplitude* A_1 for the first octave Ω_1. The amplitude intuitively corresponds to the "hardness" of the paper texture and the frequency to the size of the paper grain. Each octave Ω_i is defined from the one preceding it by multiplying the frequency by 2: $F_{i+1} = 2F_i$ and dividing the amplitude by 2: $A_{i+1} = \frac{1}{2}A_i$.

We use the method by Miné and Neyret [MN99] to implement procedural textures on graphics hardware. They use multipass texture mapping to sum a random noise texture at different scales. On modern graphics hardware, the availability of multiple texture accesses per pass can be used to avoid multiple passes. In practice, since the size of the largest-frequency octave is usually smaller than the screen, we compute this blending off-screen for one tile and store it in texture memory. All the octaves used for the texture are calculated using the same noise texture. In order to hide the grid-structure of the texture, we add a random translation to each newly created octave. An additional random rotation could further reduce regularity.

5.2 Infinite zoom using a frequency shift

In order to zoom in or out the paper, we continuously shift the frequencies of the octaves. To achieve an infinite zoom, we add new octaves in the low frequencies (zoom in) or in the high frequencies (zoom out). The addition of these frequencies is made smooth using an envelope, in practice a simple linear blend for the highest and lowest octave. Our technique is very similar to the classic audio illusion that creates a sound with an ever-ascending

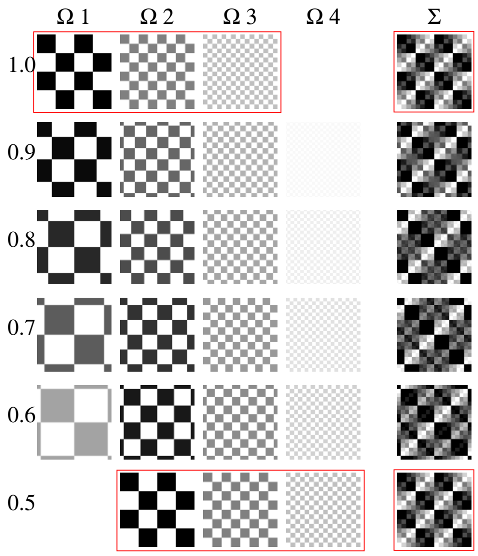


Figure 9: Illustration of infinite zoom on a checkerboard texture. The zoom factor is shown on the left. Note the inclusion of lower frequencies when the zoom factor increases. On the last line we can see that octave Ω_i and the sum Σ are the same as Ω_{i-1} and Σ of the first line (red boxes). We are thus ready to wrap-around.

or ever-descending pitch introduced by Shepard [She64]. It is also similar to the classical stairs by M.C. Escher. It is a shift in frequency space of a repetitive spectrum controlled by a frequency envelope.

When the observer moves forwards at a rate dZ/dt, the texture is scaled by a zoom factor: $zoom = zoom * \frac{D-\Delta Z}{D}$ (Eq. 4 first order approximation). This *zoom* factor *decreases* when we zoom in since a smaller portion of the image must then fill the screen (see Appendix A).

In screen space, each octave is scaled by this zoom factor, which corresponds in the frequency domain to a scale of all the frequencies of the octaves. The amplitudes have to be tuned accordingly (divided by *zoom*) in order to preserve the properties of the resulting texture.

Initially *zoom* is set to 1. During a forward translation the value of *zoom* decreases (zoom in). When *zoom* becomes equal to $\frac{1}{2}$, each octave Ω_i satisfies: $F_i(zoom = \frac{1}{2}) = \frac{1}{2}F_i(zoom = 1)$ and $A_i(zoom = \frac{1}{2}) = 2A_i(zoom = 1)$. This corresponds to a shift of the octaves.

In order to introduce new octaves in the high frequencies and to suppress the very low frequencies introduced, we shift the octaves cyclically: each octave Ω_{i+1} becomes Ω_i and the first octave becomes the last one. Smoothness is ensured by linearly blending the first and last octaves. This can be seen as our frequency envelope. The process is illustrated in Fig. 9 with a checkerboard

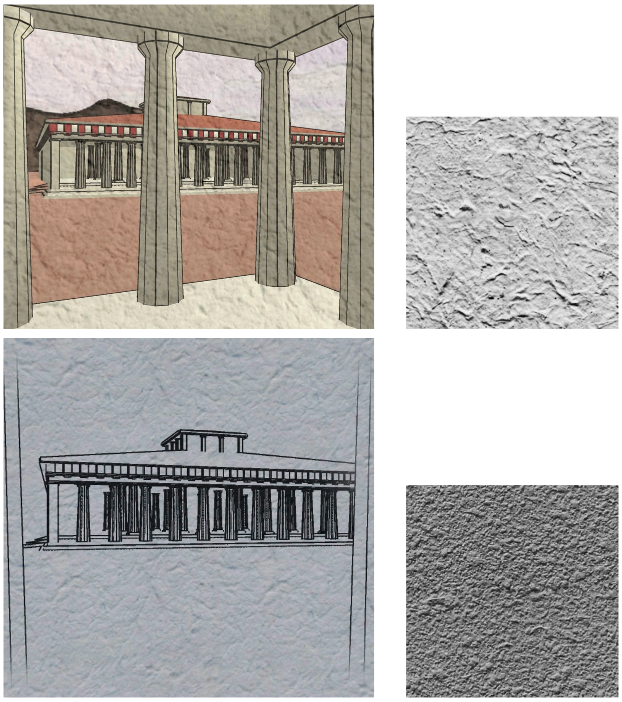


Figure 10: Different scanned papers used with our infinite zoom technique. We show a screenshot and the corresponding paper texture (with contrast enhanced for clarity.)

texture for illustration and with an actual Perlin noise. Note that the increase in the amplitude of low frequency octaves is counteracted by the frequency envelope.

At this point, we reinitialize *zoom* to 1 and we continue the forward motion in the same way. We obtain an *infinite zoom*, by compositing a fixed number of octaves. Just as in the static case, we randomly translate the newly created octave to avoid cyclic period appearance.

For backward translation, *zoom* increases and when it reaches 1, the same shift of the octaves can be applied and zoom is reset to $\frac{1}{2}$.

5.3 Using real textures

A scanned image can also be used for the octaves. We use the same technique: each octave is calculated by zooming in the original picture with a factor of 2. This permits the use of scanned paper images as shown in Figure 1 and 10.

Note that when used with the rolling-ball technique, the resulting texture may be displayed rotated on the screen. If the texture is anisotropic, the vertical direction might not be preserved. This is however a problem only with textures such as graph paper.

6 Implementation and results

The paper texture is rendered using a textured polygonal mesh. The texture coordinates are computed using the 2D transformation given by the zoom and rolling-ball technique (see Sec. 3) and modified by the spherical distortion described in Sec. 4. For each frame, we pre-render the composition of 3 to 5 octaves in a 256x256 texture tile to generate the zoomable canvas. By varying the base amplitude A_1 and lowest frequency F_1, we can modify the hardness and sharpness of the texture.

The technique described can be used with any non-photorealistic rendering method. We have implemented it in an NPR walkthrough system that uses line drawing and a simple paper-mark interaction model. When the objects are filled with color, we use a multiplicative blend between the paper color and the object color. Other paper models [LMHB00, MG02, SB00, CAS$^+$97] could also greatly benefit from our background animation. Several paper types have been tested (Perlin noise and scanned images) and provide convincing results. The scanned image needs to be periodic, and better results are obtained if it is reasonably isotropic.

The zoom depends on the subjective distance D, in order to match the canvas zoom speed to the 3D objects located at this distance. A constant value works well for walkthrough applications where no special object should stand out. The strokes slide a little on the canvas for objects at a different distance, but we did not find it disconcerting. When a special object is used from the outside in, we use the distance to the object as subjective distance. In this case, the dynamic canvas not only provides strong motion cues, but it also helps focus the attention onto the object of interest.

The method provides a dramatic improvement for NPR walkthroughs, as demonstrated by the accompanying video. The strong motion cues induced by the motion field are very effective at reinforcing the immersion. The strokes almost do not slide on the dynamic canvas, which greatly improves the impression of the picture as a whole. And because the canvas undergoes only 2D transformation, we preserve the 2D quality of the drawing.

Preliminary feedback by artists and archaeologists were very promising. As described by Strothotte et al. [SSRL96] in the context of architecture, the major advantage of non-photorealistic rendering for archaeological walkthroughs is that it emphasize that the displayed scene is only an hypothesis. An NPR walkthroughs allows archaeologists to use both the strength of 3D graphics for model exploration and their traditional drawing style. In this context, our dynamic canvas provides strong motion cues and reinforces the 3D immersion, while respecting the traditional 2D qualities of the drawing.

7 Conclusions and future work

We have presented a method that animates the background canvas for NPR walkthroughs. Using simple 2D transformations and a spherical distortion, we approximate the complex motion field produced by a 3D displacement. This provides compelling motion cues and greatly improves the "immersive" impression for NPR walkthroughs.

This work opens several directions of future research. We are working on more advanced paper-stroke models. The subjective distance could be computed on the fly in an "autofocus" fashion. The choice of paper motion might be influenced by the medium used for the marks. For example, some marks are mostly opaque (e.g., oil painting) and the background canvas might be visible only in some places.

The rolling-ball metaphor could also be used to create multiperspective panoramas [WFH+97]. In the case of off-line animation, the optimization approach by Wood et al. could be applied to the motion of the background texture to obtain a better approximation. More generally, we believe that the framework of motion fields and their approximation with 2D transform has great potential for NPR animation. We are planning to apply it to marks such as brush or pencil strokes, and also to larger regions such as watercolor effects. In these cases, the problem is more complex because marks need to be more strongly attached to the depicted 3D objects.

Acknowledgments

This research was funded in part by the INRIA Action de Recherche Coopérative ARCHEOS (http://www-sop.inria.fr/reves/Archeos).

Thanks to Fabrice Neyret and Samuel Hornus for numerous discussions and observations. Thanks to Ray Jones, Sara Su and Victor Ostromoukhov for proofreading the paper.

References

[Bar88] Michael Barnsley. *Fractals Everywhere*. Academic Press, 1988.

[BK00] S. Baker and T. Kanade. Limits on super-resolution and how to break them. In *Proc. of CVPR*, 2000.

[CAS+97] C. Curtis, S. Anderson, J. Seims, K. Fleischer, and D. Salesin. Computer-generated watercolor. *Proc. SIGGRAPH*, 1997.

[Cur98] C. Curtis. Loose and Sketchy Animation. In *SIGGRAPH: Conf. Abstracts and Applications*, 1998.

[Dan99] Eric Daniels. Deep canvas in disney's tarzan. In *ACM SiGGRAPH sketch and applications*, 1999.

[Den96] Borden D. Dent. *Cartography: Thematic Map Design*. WCB/McGraw-Hill, 4th edition, 1996.

[Dur02] F. Durand. An invitation to discuss computer depiction. In *Symposium on Non-Photorealistic Animation and Rendering (NPAR)*, 2002.

[Fre65] A. P. French. *Newtonian Mechanics*. W. W. Norton Company, 1965.

[GG01] Gooch and Gooch. *Non-Photorealistic Rendering*. AK-Peters, 2001.

[Gib79] J. J. Gibson. *The Ecological Approach to Visual Perception*. Houghton Mifflin, 1979.

[Hor86] B. P Horn. *Robot Vision*. MIT Press, Cambridge 1986, 1986.

[HP00] A. Hertzmann and K. Perlin. Painterly rendering for video and interaction. In *Non-Photorealistic Animation and Rendering*, 2000.

[Hut81] J. Hutchinson. Fractals and self-similarity. *Indiana Univ. J. of Mathematics*, 30:713–747, 1981.

[KKL+00] A. Klein, M. Kazhdan, W. Li, W. Toledo Corra, A. Finkelstein, and T.Funkhouser. Non-photorealistic virtual environments. *Proc. SIGGRAPH*, 2000.

[Lit97] P. Litwinowicz. Processing images and video for an impressionist effect. *Proc. SIGGRAPH*, 1997.

[LMHB00] A. Lake, C. Marshall, M. Harris, and M. Blackstein. Stylized rendering techniques for scalable real-time 3d animation. In *Non-Photorealistic Animation and Rendering*, 2000.

[Mei96] B. Meier. Painterly rendering for animation. *Proc. SIGGRAPH*, 1996.

[MG02] A. Majumder and M. Gopi. Hardware accelerated real time charcoal rendering. In *NPAR 2002: Symposium on Non Photorealistic Animation and Rendering*, June 2002.

[MN99] A. Miné and F. Neyret. Perlin textures in real time using OpenGL. Technical report, RR-3713, INRIA, 1999. http://www-imagis.imag.fr/Membres/Fabrice.Neyret/publis/RR-3713-eng.html.

[Per85] Ken Perlin. An image synthesizer. In *Computer Graphics (Proc. SIGGRAPH 85)*, volume 19, pages 287–296, July 1985.

[PHMF01] E. Praun, H. Hoppe, M., and A. Finkelstein. Real-time hatching. *Proc. SIGGRAPH*, 2001.

[Rey02] C. Reynolds. Stylized depiction in computer graphics non-photorealistic, painterly and 'toon rendering. http://www.red3d.com/cwr/npr/, 2002.

[SB00] M. Costa Sousa and J. Buchanan. Observational models of graphite pencil materials. *Computer Graphics Forum*, 19(1):27–49, March 2000.

[She64] R. Shepard. Circularity in judgments of relative pitch. *J. Acoust. Soc. Am.*, 36:2346–2353, 1964.

[SS02] Thomas Strothotte and Stefan Schlechtweg. *Non-Photorealistic Computer Graphics. Modeling, Rendering, and Animation*. Morgan Kaufmann, San Francisco, 2002.

[SSRL96] J. Schumann, T. Strothotte, A. Raab, and S. Laser. Assessing the effect of non-photorealistic rendered images in cad. In *Proceedings of ACM CHI*, 1996.

[WFH+97] D. Wood, A. Finkelstein, J. Hughes, C. Thayer, and D. Salesin. Multiperspective panoramas for cel animation. *Proc. SIGGRAPH*, 1997.

[WPFH02] M. Webb, E. Praun, A. Finkelstein, and H. Hoppe. Fine tone control in hardware hatching. In *NPAR 2002: Symposium on Non Photorealistic Animation and Rendering*, 2002.

A Texture space transformation from 3D rotation

We describe the 2D transformation that approximates a 3D camera rotation. This rotation is defined by an instantaneous axis Ω and an incremental angle $\Delta\alpha$.

The paper texture coordinates are in the range $[-zoom, zoom]$ for the vertical direction, and are scaled by the screen aspect ratio in the horizontal direction. The radius of the rolling ball is then defined by:

$$R = \frac{zoom}{\tan(\frac{1}{2} fov)}$$

where fov is the camera vertical field of view.

We first consider the case where Ω is orthogonal to the viewing direction (Fig. 4). The texture coordinates then have to be translated in a direction orthogonal to Ω, by an offset $\delta = R\,\Delta\alpha$, the distance made by the rolling ball.

In most applications, the camera is rotated around its own X and Y axis and the previous equation is sufficient. However, if one wants for instance to maintain the horizon horizontal in a walkthrough application, the camera will be rotated around a fixed up vector. We introduce the elevation angle α around the camera X axis and the rotation angle β around the world vertical axis (see Fig. 11a).

In this general case, Ω intersects the texture plane at a point C, which is the instantaneous rotation center of the texture coordinates (see Fig. 5). The screen center then describes a cone when the camera rotates around the vertical up vector.

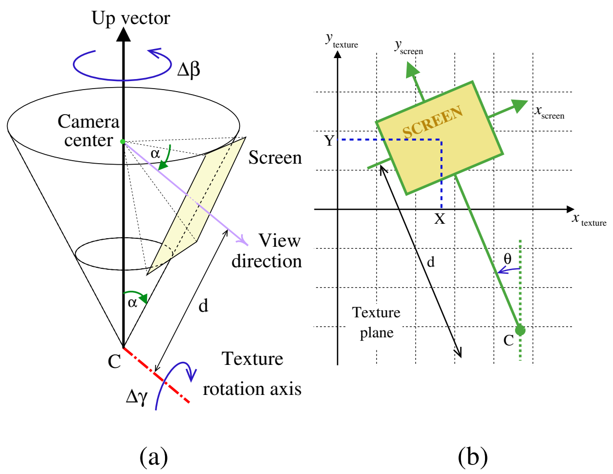


Figure 11: (a) Geometric configuration of the camera movements : cone described by the screen during a $\Delta\beta$ rotation. (b) Geometric configuration of the texture plane.

In the screen-texture plane, a world $\Delta\beta$ rotation will result in a rotation around C, with an angle $\Delta\gamma$:

$$\Delta\gamma = \sin(\alpha)\Delta\beta \qquad\qquad d = \frac{R}{\tan(\alpha)}$$

In our walkthrough application, we combine a camera X axis up-down rotation ($\Delta\alpha$), with a vertical axis panning rotation ($\Delta\beta$). The paper texture rectangular window is then defined by its center (X, Y) and its orientation γ (Fig. 11b), updated as follows:

$$\begin{pmatrix} X \\ Y \end{pmatrix} \mathrel{+}= \mathcal{R}_\gamma \begin{pmatrix} d\ \sin\Delta\gamma \\ d\ (1-\cos\Delta\gamma)\ +\ R\,\Delta\alpha \end{pmatrix} \qquad \gamma \mathrel{+}= \Delta\gamma$$

where $\mathcal{R}_\gamma$ is the rotation of angle γ in the texture plane.

Note that this equation is still valid when $\alpha = 0$, where it reduces to $\begin{pmatrix} X \\ Y \end{pmatrix} \mathrel{+}= \mathcal{R}_\gamma \begin{pmatrix} R\,\Delta\beta \\ R\,\Delta\alpha \end{pmatrix}$, a simple rotation around the camera X and Y axis.

If a is the camera aspect ratio, the final rectangular texture coordinates are: $\begin{pmatrix} X \\ Y \end{pmatrix} + \mathcal{R}_\gamma \begin{pmatrix} \pm zoom\ a \\ \pm zoom \end{pmatrix}$.

B Spherical warp equations

The first step of the spherical deformation algorithm (see Sec. 4) projects a point of the screen onto the sphere. These spherical coordinates (θ, φ) are measured with respect to the up-vector, so that for the center of the screen $\theta = 0$ and $\varphi = \alpha$ (Fig. 12a).

The sphere parallel for φ is projected in the texture plane onto a circle of center C, and radius defined by the offset o relative to the projection of the center of the screen (see Fig. 12b). In order to ensure no sliding along the screen vertical center line for up-down rotation , o has to satisfy $o = (\varphi - \alpha)\,R$.

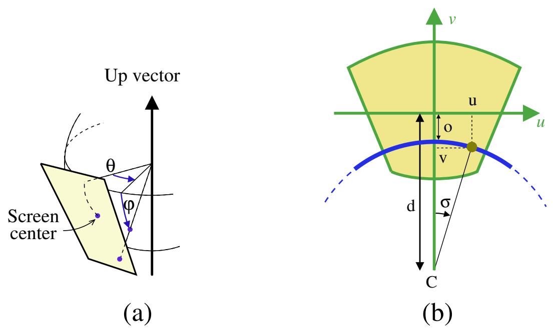


Figure 12: Spherical and texture plane coordinates

Once the texture circle has been determined, the (u, v) texture coordinates of the screen point are determined by the angle σ (Fig. 12b). This angle is such that when the sphere rotates, the distance covered on the texture circle matches the arc length on the sphere $(d-o)\,\sigma = R\cos(\varphi)\,\theta$. Finally we get:

$$u = (d-o)\ \sin(\sigma) \qquad\qquad v = o\ +\ (d-o)(1-\cos(\sigma))$$