Dynamic Texturing of Botanical Environments

Andrew A. Cove
acove@andrew.cmu.edu
Advisor: Doug L. James

Carnegie Mellon University

Figure 1: **Irises:** A large field of irises blowing in the wind.

Abstract

Animating realistic large botanical environments for real-time applications is computationally intractable given the cost of nonlinear dynamics, collisions, and wind coupling, as well as global illumination. However, by combining data-driven animation techniques with three dimensional packing techniques, it is possible to produce expansive and immersive interactive botanical worlds. Data-driven animation techniques enable real-time synthesis of physically based motions without the cost of run-time simulation. The overhead costs of precomputing motion and lighting for a variety of objects, behaviors, and environmental conditions are large, but tolerable given the performance benefits at run-time. Since a primary goal of physically based modeling is to synthesize real world environments, a natural extension of precomputing and synthesizing complex behaviors for simulated objects is to place the objects in scenes representative of their natural environments. However, the cost of assembling and precomputing physical states for an environment scales as the environment grows in size and complexity, and the reuseability of the data diminishes. We provide techniques for synthesizing expansive, interactive botanical environments that maintain the low run-time costs of data-driven animation without incurring the overhead costs of building and simulating large scenes. We utilize a small library of data-driven irises and an extension of stochastic methods for tiling Wang Tiles to generate a field of over 40,000 irises that can be animated in real-time under varying wind and lighting conditions.

1 Introduction

In this paper, we propose a method for creating large environments populated with realistically behaving plants. Each plant in an environment is an instance drawn from a small library of plant mo-

tions and lighting data. By populating the environment with plant instances, we produce realistic, dynamic botanical scenes without incurring the costs associated with physical simulation and global illumination of complex environments. We animate the plant instances with a data-driven animation technique that draws on both video textures [Schödl et al. 2000] and motion graphs [Kovar et al. 2002], and determine the position for each plant by extending the usage of Wang Tiles described in [Cohen et al. 2003]. We also describe the implementation of our techniques in the form of an iris field, and explain some of the optimizations that we make to enable the field to be explored interactively and to respond to dynamic wind and lighting. Finally, we recommend future areas of study to improve the physical accuracy and visual quality of our methods.

1.1 Our Contribution

We present a method to build expansive, dynamic botanical environments. We construct an environment using instances from a library of motion and lighting data, allowing the plants in the environment to respond physically to changes in wind and lighting, with low run-time cost facilitating interactive exploration of the scene. In addition, we propose an extension to Wang Tiles [Cohen et al. 2003] that accommodates the use of deformable models without decreasing packing density.

1.2 Related Work

Our work incorporates aspects of three different areas of computer graphics to produce large scale, complex botanical environments. Data-driven animation techniques enable the irises in our field to respond physically to dynamic wind conditions. We light our scene with a real-time approximation to global illumination, so that the iris field exhibits the appropriate response to changing lighting conditions. To populate the field, we distribute the iris clumps according to a technique for three dimensional space packing as well as non-periodic texture generation.

We animate the plant instances in our environment by applying the principles of video textures [Schödl et al. 2000] to animations generated from motion graphs, originally described in [Kovar et al. 2002]. Motion graphs represent motion synthesis as graph walks on a directed graph, in which edges represent either original motion data or automatically generated transitions. Walking the graph generates a continuous series of motion by playing back original motion and smoothly transitioning to another data set. [James and Fatahalian 2003] extend motion graphs by using impulse response databases for deformable objects to transition between impulsive motion clips. Our plants are animated by impulsive motion clips, similar to those in [James and Fatahalian 2003]; however, since we do not have direct physical interaction with our plants, continuous plant motions are essentially played back as a series of animation clips, connected by transition points with similar characteristics, as in video textures. However, the motion stored in the motion graphs for our plants is indexed according to varying wind levels. As such, our motion texturing responds dynamically to input by preferring transitions in the graph that best match the given input conditions.

Our plants are lit using appearance models based on a low rank approximation to the diffuse radiance transfer global illumination model for low-frequency lighting [James and Fatahalian 2003], [Sloan et al. 2002]. In addition, we use the same precomputed radiance transfer technique to approximate shadowing effects on the ground plane.

[Cohen et al. 2003] use Wang Tiles to generate non-periodic tilings for non-repeating textures and to position geometry in large scenes. They provide a botanical scene composed of a large collection of sunflowers, presented as static geometry and rendered as layered depth images. We present extensions to their methods for the purpose of generating environments that contain dynamic models, to be rendered in real-time as three dimensional geometry.

Our work is very similar to that of [Perbet and Cani 2001]. They produce a large scale, interactive prairie through the use of procedurally animated grass primitives and texture-based level of detail. However, their work is better suited to situations where inter-primitive collisions are less visible. Additionally, creating the grass primitives does not carry the modeling or simulation costs associated with our plant models.

2 Details of Approach

The limitations of building an expansive, densely packed botanical environment present themselves immediately. The modeling cost alone of a field of thousands of complex plants is an unmanageable burden. However, given such an environment, the insurmountable hurdle is animating and displaying the world interactively. The complexity of global-illumination, simulation, and collision resolution of a large scale, hand crafted environment is too great to be computed at run-time. To facilitate the generation and presentation of a botanical environment comprised of tens of thousands of dynamically responsive plants, we solve the computational challenges for a small set of plant primitives, and use these primitives as building blocks with which to construct an environment with the desired realism and complexity, with manageable run-time costs that allow us to interact with the world in real-time.

2.1 Instancing

We use a set of four iris "clumps," shown in Figure 2, as the building blocks for our iris field. Each clump contains six to seven irises in close contact. For each clump, we generate a motion library by simulating the clumps in a black-box Navier-Stokes wind simulator, with collisions resolved using penalty forces. The data is stored in a reduced coordinate representation as a motion graph, indexed by wind level, to be used to generate motion at run-time. For each clump, we also precompute radiance transfer data, which we use at run-time to compute the appearance of the clumps under dynamic lighting conditions. To reduce the visual redundancy inherent in reusing the clumps thousands of times throughout the scene, we give each instance of a clump a different initial index into the motion graph. Thus, even though the animation is generated from the same set of motions, the clumps will follow different paths according to the seed and the randomly chosen transitions.

Figure 2: **Iris Clumps:** The four iris clump primitives instanced throughout the iris field.

Building the iris field out of clump instances reduces the overhead of creating the environment. If the full field were treated as a single object when simulating motion and computing global-illumination data, computation times would be intractable. Managing the complex interactions of the numerous plants in the field

requires the resolution of numerous collisions and adds significant complexity to the effects each plant has on the appearance of its neighbors. In comparison, instancing has two primary disadvantages. First of all, since collisions are handled during simulation, each clump's motion graph only contains behavior for collisions between the irises within the clump. A full collision detection and resolution system would have to execute at run time to prevent inter-clump collisions. We alleviate this disadvantage by carefully positioning the iris clumps so that they can not possibly interact with each other, as described in the next section. Second, the individual clumps will have inaccurate illumination data at their boundaries, because the appearance model is generated without knowledge of the positions of neighboring clumps. Although the clumps' lighting data can be computed with additional geometry present to provide occlusion and diffuse inter-reflection effects, the lighting for each instance will not be correct for its actual neighbors. This disadvantage could be eliminated by recomputing the appearance models for each clump at each instance of the clump on each base Wang Tile. While this method provides the correct lighting for each clump within the base tiles, it does not solve the problem entirely, because of the effects that plants in neighboring tiles have on the clumps at each of the base tiles' borders. We could endeavor to produce appearance models for each clump for each base tile for each possible neighbor, but the problem grows exponentially while reducing the positive benefits of instancing to save preprocessing time. Although the use of instances reduces the entropy in the environment and provides only an approximation to global illumination, the improvement in initialization and run-time costs makes the deficiencies tolerable.

2.2 3D Packing Problem

Having determined the building blocks with which to construct the environment, we are left with the challenge of arranging the world out of clump instances. In order to appear natural, the instance positions must maintain the invariant of physicality - plant clumps can not overlap and penetrate each other - and must not contain unnatural visible patterns - for example, a grid of irises would not be particularly natural.

Wang Tiles can be used to reduce the visual repetition in a large scene while preventing visual discontinuity across edges and maintaining low costs to populate the environment [Cohen et al. 2003]. Each Wang Tile is a square in which each edge is assigned a color. In a tiling pattern, all shared edges between neighboring tiles must have matching colors. Cohen et al. present a process involving two stochastic methods to create an environment out of Wang Tiles. They build a pattern by sequentially placing tiles. At each step, a tile is randomly selected from the set of tiles whose edges match the open edges of the current configuration. Since multiple tiles share the same colors for individual edges, the stochastic process should not repeat the same patterns. An example Wang Tile pattern is shown in Figure 3.

The other stochastic method in [Cohen et al. 2003] is used to populate each Wang Tile. The authors place objects on to the tiles by throwing darts according to a Poisson disc distribution, depicted in Figure 4. At every iteration, a random position is selected on a tile. The radius of the object to be placed defines a disc around the dart's position. If the dart's disc overlaps the boundaries of any objects already on the tile, the dart is rejected. Additionally, if the dart's boundary circle overlaps any of the tile's edges, the boundary circle is tested against all of the possible neighbors across the overlapped edges, to prevent discontinuities in the final tiling. If the dart does not intersect any of its own tile's other darts, or any possible neighbors' darts, the object is added to the tile at the dart's position. Since the position is selected at random, there is no repetition within the tile. However, if a full tiling is viewed from

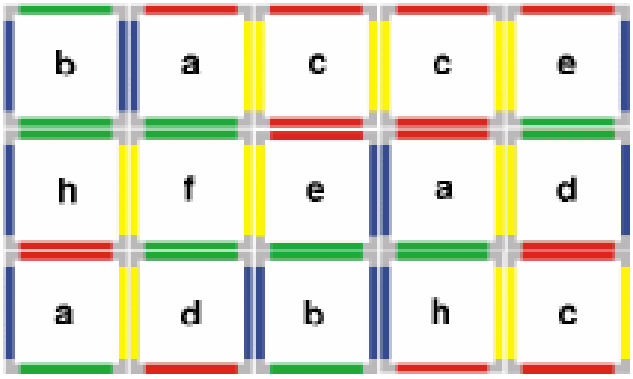

Figure 3: **Wang Tiles:** A tile pattern composed of randomly selected tiles with matching edge colors [Cohen et al. 2003].

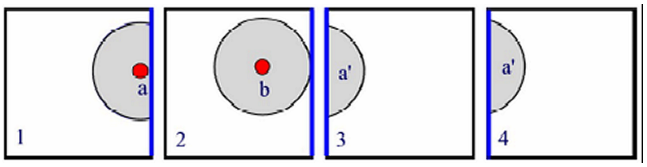

Figure 4: **Wang Tile Population:** Dart throwing using poisson-disk distribution [Cohen et al. 2003].

a distance, it may be possible to recognize the features within an individual tile at the tile's various positions within the pattern.

The method to generate patterns of Wang Tiles presented in [Cohen et al. 2003] does not lend itself to certain features of the iris field. We want the iris field to be densely populated and to be animated. However, the use of the Poisson disc distribution of dart positions can not satisfy both goals. Animating the clumps, particularly at high wind levels, greatly increases the radius of the boundary circle around each clump's two dimensional swept area. Testing dart positions with the larger boundary circle is necessary to prevent visual artifacts due to interpenetration; however, as the darts' radii grow, the amount of empty space between objects in the individual tiles also grows, and the overall plant density drops. We alleviate this problem by replacing the disc with a 3D volume. We sample each clump into a uniform spatial grid. In addition, we add samples of the clumps at various displacements across wind levels (with an emphasis on the highest wind level). Rather than maintaining the Poisson disc distribution, we test the cell overlaps of the voxelizations associated with each dart. By testing against the swept volumes, we are able to produce much denser plant packings within each Wang Tile without worrying about collisions between clumps. Combining the collision free, random positioning of plants within each Wang Tile and the non-repeating tile pattern produced by edge color matching, we generate an iris field that appears natural without the cost of positioning every clump instance throughout the field. Figures 5 and 6 show a Wang Tile populated by voxel-based dart throwing and the same tile drawn with the actual iris clumps.

2.3 Implementation and Optimization

The primary goal of our work is to produce expansive, interactive environments. In order to demonstrate the effectiveness of instancing from the motion library to populate scenes, the scene must be densely populated and be responsive at interactive rates, without compromising size or complexity. To achieve the performance lev-

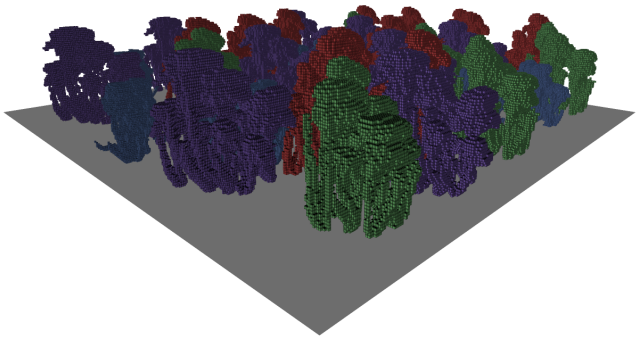

Figure 5: **Voxelized Tile Population:** A densely packed Wang Tile, generated by testing the voxel overlap of neighboring plant clumps for each dart throw.

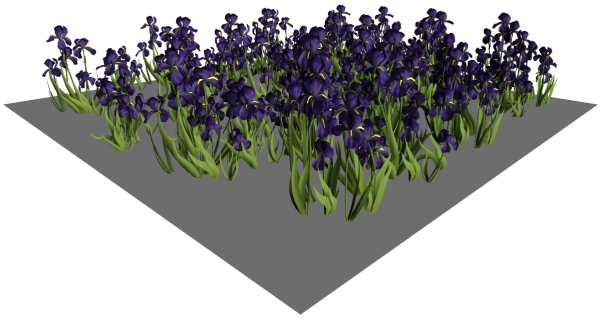

Figure 6: **Tile Population:** The same Wang Tile, displayed as actual plant clumps.

Figure 7: **Levels of Detail:** Geometry for an increasingly reduced plant clump mesh.

els necessary to allow dynamic camera changes and varying wind levels, we incorporate level of detail versions of the clumps and take advantage of programmable graphics hardware and careful optimization of our OpenGL implementation.

To reduce the cost of animating and rendering a full field of iris clumps, we create five levels of detail for each clump. Clumps close to the camera are shown at their full, first level of detail; the plants are animated using their full motion libraries and lighting data. Because of the high dimensionality of the motion data, the deformation for the full detail plant models must be computed in software. As the distance between a clump and the camera grows, we substitute increasingly less expensive level of detail models for the clump. The subsequent levels of detail use a low-rank matrix approximation to the full reduced-coordinate motion data. The geometry for each clump is reduced using a quadric-based reduction method, enhanced to appropriately weight the lighting and motion data of the mesh as the number of vertices is reduced. Figure 7 depicts levels of reduced geometry for an individual clump. Each level of detail for each clump is stored in a display list, with the lighting and motion data stored as vertex properties. We animate the lower level of detail plants by passing an index vector for the motion data as a parameter to a vertex program, which computes the per-vertex displacements to deform the vertices. Because the per-vertex deformations are done in parallel on the graphics card, animating the numerous lower detail models is computationally inexpensive. The low-rank motion approximation does not significantly decrease the visual quality of the overall scene because the reduced-rank motion is only used far from the camera.

To achieve our performance goals, we heavily optimized our OpenGL code to reduce the amount of calls to the OpenGL library and the amount of data sent to the video card every frame. Passing the geometry to OpenGL as triangle-stripped vertex arrays, rather than as an arbitrary ordering of triangles of individual vertices, incurred a three times performance improvement. We reduced the penalty for switching textures by building a texture atlas of all the textures for the irises. By packing all of the textures into a single texture and appropriately offsetting vertex coordinates, we can render all of the irises in the field without having to change textures.

The order in which we render the iris field geometry has a significant impact on performance. The organizational structure that we use to generate the environment, as a Wang Tile pattern, is not conducive to efficient rendering of full three dimensional geometry. The performance of the rendering pipeline was not an issue in [Cohen et al. 2003], because the environment was rendered as a layered depth image. However, displaying our iris field by iterating through each tile in the pattern, and then iterating through each plant in the tile, hurts performance by repeatedly changing the geometry for the card to render. Thus, our run-time implementation contains no notion of the Wang Tiles, and instead maintains a list of every instance for each clump, and each frame renders all visible

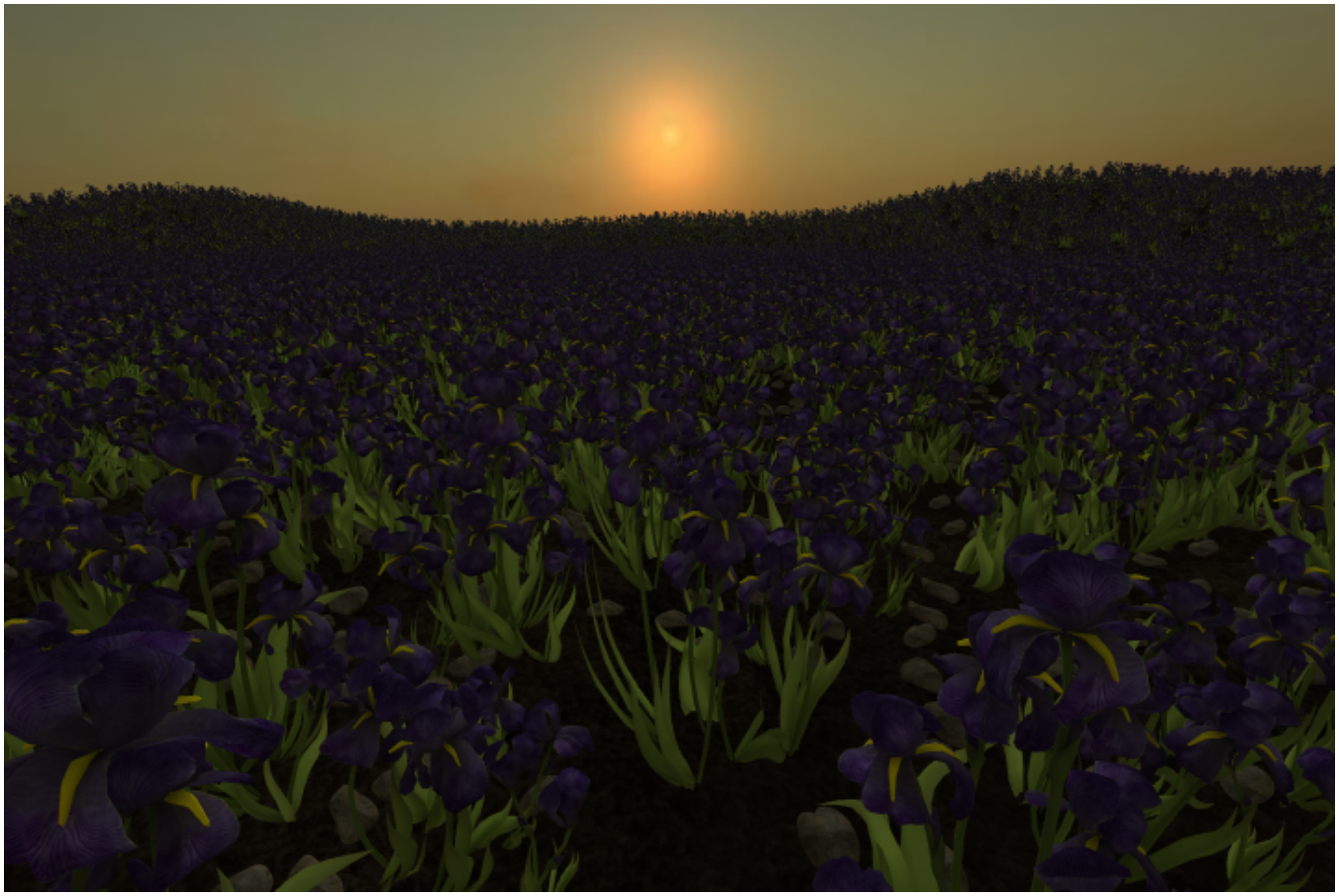

Figure 8: **Iris Field at Sunrise:** The iris field changes appearance as the sun rises and sets over the course of a day.

instances of a particular clump before moving on to the next. This, together with the use of texture atlases, allows us to change as little data on the video card as possible when drawing each frame.

3 Iris Field

3.1 Construction Details

The iris field in our video is comprised of a fifteen by fifteen grid of tiles built from a base set of eight Wang Tiles. The plant positions on each tile are generated by dart throws, with voxel-based overlap testing. The tiling pattern was created as outlined in [Cohen et al. 2003]. The hills in the distance and the associated displacement for each clump were calculated using bilinear interpolation of a very low-frequency heightmap. The vertical displacement of each clump creates a risk of interpenetration between neighbors because there is no knowledge of the vertical position of a clump when the base Wang Tiles are being populated. However, since the hills serve primarily as distant background, the camera is too far away for artifacts to be noticeable. We recommend a solution for this problem in our discussion of future work. There are approximately 7,700 clumps in the field (about 46,000 irises). In addition, there are thousands of rock instances that were placed on the base Wang Tiles in the same fashion as the clumps.

To demonstrate that the iris field is responsive to dynamic changes in lighting, we display the field over the course of a full day. The skies in the full day and individual time sequences are from a sequence of 1,000 sky images. We use alpha blending to smooth the transition between sky textures, since we use multiple frames of motion per sky image. In addition, we calculate a spherical harmonic light vector for the lighting in each sky frame, and apply the precomputed radiance data to the light vector to compute the diffuse radiance of the clumps. As we change sky frames, we update the lighting for the clumps, the rocks, and the ground plane. Figures 1 and 8 highlight the effects of varying lighting conditions.

3.2 Run-Time

The iris field is implemented in Java, using the GL4Java OpenGL implementation. The full implementation with 46,000 irises, changing sky textures and lighting, and a few thousand rocks, runs at more than 1 frame per second. Without the rocks and with the sky texture held constant, the scene runs at about 5 frames per second. The memory usage is almost 1.5 GB, comprised mostly of motion and lighting data. The iris field can be navigated with an interactive camera, and any part of the field can be reached. As is shown in the video, the camera is not limited to specific types of views or positions. The level of detail of each instance changes dynamically based on the instance's distance from the camera. Since the level of detail motion and the full motion data use the same index from the motion graphs, the lower detail clumps and full clumps are similar enough that the changes in levels of detail do not create visual artifacts.

A simple wind generator sends waves of wind across the field. Each plant has motion data for four wind levels (with the wind blowing in a consistent direction). The clumps respond dynami-

cally to the changing wind speeds, moving to the appropriate paths in the motion graphs for the higher wind levels, and returning to low-wind motion when the waves have passed.

4 Conclusions

In this paper, we presented a framework for constructing expansive, complex interactive botanical environments using instances of a small set of plant clumps with libraries of precomputed motion and lighting data. We use instancing to reduce the modeling and simulation overhead inherent in creating and animating a full field of unique irises with physically-based motion, and apply concepts of video texturing to a motion graph of impulsive motion clips to animate each instance. Additionally, we extend the use of Wang Tiles to utilize voxel-based boundary testing to accomodate model deformations. We present our work in the form of an iris field, containing 46,000 irises that respond to dynamic wind and lighting conditions, which can be explored interactively.

4.1 Future Work

The challenge of lighting each iris clump in a method that considers the influence of neighboring clumps is daunting. Although the aforementioned method of computing radiance transfer data by tile instead of by clump, and additionally considering all possible neighboring tiles, would be a correct solution for the problem, the costs do not scale well with the number of base Wang Tiles and the number of plants in each tile. A better solution would represent the influence of neighboring geometry in the light vector used as an index into the precomputed radiance transfer data, so that the output radiance would take the effects of neighbors into account.

For our framework to scale to environments with no restrictions on camera motion, the risk of inconsistency as a result of vertically shifting the clumps in a tile must be eliminated. The most direct solution to this problem is to remove the use of Wang Tiles to generate our environment. Although building the tiling pattern from the set of base tiles reduces the cost of distributing plants, the ignorance of the base tiles to vertical changes in the actual tiling prohibits a guarantee against interpenetration in anything but flat environments. Instead, the positions of clump instances can be generated by throwing darts over the full environment. Although this method will be slower (in the case of our iris field, darts will be thrown at 225 tiles instead of 8 tiles), the consistency of the physicality of the field should not be compromised.

References

COHEN, M. F., SHADE, J., HILLER, S., AND DEUSSEN, O. 2003. Wang tiles for image and texture generation. *ACM Transactions on Graphics 22*, 3 (July), 287–294.

JAMES, D. L., AND FATAHALIAN, K. 2003. Precomputing interactive dynamic deformable scenes. *ACM Transactions on Graphics 22*, 3 (July), 879–887.

KOVAR, L., GLEICHER, M., AND PIGHIN, F. 2002. Motion graphs. *ACM Transactions on Graphics 21*, 3 (July), 473–482.

PERBET, F., AND CANI, M.-P. 2001. Animating prairies in real-time. In *2001 ACM Symposium on Interactive 3D Graphics*, 103–110.

SCHÖDL, A., SZELISKI, R., SALESIN, D. H., AND ESSA, I. 2000. Video textures. In *Proceedings of ACM SIGGRAPH 2000*, Computer Graphics Proceedings, Annual Conference Series, 489–498.

SLOAN, P.-P., KAUTZ, J., AND SNYDER, J. 2002. Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. *ACM Transactions on Graphics 21*, 3 (July), 527–536.

Figure 9: **Iris Field:** Another view of the iris field.