EG UK Theory and Practice of Computer Graphics (2005)
L. Lever, M. McDerby (Editors)

Arachnid Simulation : Scaling Arbitrary Surfaces

L. ap Cenydd[†] and W. Teahan[‡]

School of Informatics, University of Wales Bangor, UK

Abstract

There has been little research done into the realistic simulation of creatures with the ability to crawl across arbitrary surfaces, clamber up walls and walk across ceilings. Realistic simulation of such feats would be of benefit to fields such as arthropod phobia therapy, the animation of computer game characters and Artificial Life research.

We have implemented a system that can produce real-time simulation of a spider traversing across an arbitrary surface. The simulation uses a combination of a behavioral system, an orientation system, a procedural gait generator and an inverse kinematics solver to produce the real-time dynamic animation.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Three-Dimensional Graphics and Realism-Animation; I.6.8 [Simulation and Modeling]: Types of Simulation-Animation

1. Introduction

Many animals, such as lizards and arthropods have the ability to easily clamber up walls or even walk across ceilings. However, there has been little research into realistic simulation of such feats. This paper proposes an approach to animating creatures crawling across arbitrary surfaces which is both dynamic and visually realistic.

Possible application areas of such a simulation include arthropod phobia therapy, where a dynamic virtual creature capable of unbounded movement would be a benefit to advanced stages of treatment. Another is the animation of computer game characters, where an increased interaction and unrestricted movement with the environment would enhance realism and immersion. Finally, the ability to explore the surface of the environment in such an unrestricted way allows for artificial life research into enhancing the behavior and cognition of the creature.

The rest of the paper is as follows. Section 2 deals with work related to ours, section 3 explains the motion and behavioral systems of our simulation, section 4 provides detail on our spider model and animation system and section 5 details our current implementation.

[†] Email : llyr@informatics.bangor.ac.uk
[‡] Email : wjt@informatics.bangor.ac.uk

2. Related work

Dynamic animation of articulated characters is far from a trivial problem, and there has been a great deal of research into issues concerning legged locomotion and landscape alignment which bear relevance to our work.

2.1. Legged locomotion

Automatic gait generation is concerned with producing realistic walking gaits for simulated creatures. An excellent survey of work related to both physically and kinematics based human gait generation can be found in [MFCD99]. Common issues of relevance to our work includes gait adjustment in walking along an arbitrary path and step adjustment over uneven terrain. Sun et al. [SM01] addressed these two issues using three modular components, allowing for curved path locomotion and uneven terrain locomotion using sagittal elevation angles and an inverse motion interpolation algorithm.

For arthropod gait generation, which we are primarily concerned with, much research has been undertaken in the fields of biology [Bow77] [Bow81] and robotics [Fer95] [BQCR97]. Whilst most of this research is not directly applicable to our work, there are some issues of generating hexapod and octopod gaits which are universal. These include both the detail of the base walking gait, and the dynamic adjustment to the walk cycle due to changes in the creature's speed and heading necessary to keep all limbs in sync.

Parker [Par98] used cyclic genetic algorithms to generate a variety of arachnid walking gaits. Parker tested and developed the evolved stable walking gaits using a physical based model of an eight legged robot.

McKenna and Zeltzer [MZ90] developed a dynamic locomotion simulation of a hexapod, where the coordination of a kinematic cockroach is automatically generated by a gait controller, with a step and stance motor programs applying appropriate forces to the limbs.

Cruse et al. [CDK*99] used artificial neural networks to investigate insect walking. The system consists of modules (or agents), one for each leg. Each leg controller is further constructed by a number of agents, one for the control of swing movements, one for stance, a selector net and several feedforward nets providing sensory input. The authors assume that it is this decentralization of the control scheme that leads to a flexible system.

For octopod locomotion simulation, Klaassen et al. [KLSK01] developed a biomimetic control scheme for a walking robot. The joints in each of the robot's legs provide three degrees of freedom, and are driven by an alternating tetrapod walking gait which is almost exclusively the gait of real arachnids. Their approach is based on two biological control principles, the Central Pattern Generator (CPG) and the Reflex. The two principles are combined with Basic Motion Patterns (BMPs), which describe the trajectory of a leg during a stride. Under normal circumstances the robot's legs are controlled by the CPGs, which represent a low-level, hard-wired locomotion scheme specific for the species. Reflexes are only activated by exceptions, such as an obstacle blocking the leg's path.

2.2. Terrain following

Steed [Ste97] developed a system for efficient navigation of a virtual environment where the technique used can track the surface point an object is above at a small computational cost. The algorithm used to determine the surface point takes the current viewpoint and the projected next viewpoint (extrapolated from the character's current position and direction) and returns an appropriate position where there is no environment collision or deviations in the given height above the surface.

The next position is found by traversing a cell structure, based on a winged edge data structure, that contains the information required to determine surface heights and surface collisions in the edges of the cells.

In QOTA (Quick Oriented Terrain Algorithm) [BW97], Barrus and Waters developed a fast algorithm for terrain following in virtual environments. The algorithm focuses solely on the problem of intersecting lines of a predetermined orientation (projected from an object or character) with a terrain model. QOTA uses a pre-processing step which sorts the terrain's polygons into a quadtree, adding bounding boxes and polygon edge equation parameters to speed up polygon containment checking. By consulting the quadtree, the algorithm is capable of locating the intersection point very quickly.

2.3. Scaling arbitrary surfaces

As far as we are aware, there has been no attempt to combine the techniques outlined in this section for the simulation of creatures capable of scaling arbitrary surfaces in virtual environments.

Current realtime techniques, especially prevalent in the computer game industry, are usually comprised of simply rotating the standard walking animation of the creature in question from one orientation to another, as it moves from once surface to another. At best, a transitional animation is blended between large orientation changes. A compromise inherent to this sort of technique sees the creature move very quickly or sporadically, so as to not draw attention to the limits of the technique. Furthermore, the environments created to house these creatures are also based around the limitations of such a technique.

3. Traversing the surface of an object

This section details the important features of the motion and behavior system we have developed.

3.1. Moving around

The motion system, which controls the global position of the creature is based on a simple vehicle model, similar to that proposed by Craig Reynolds [Rey99]. The creature has at any time a single overall goal – in our current implementation this goal is to randomly explore the surface of the environment. The goal is decomposed into a series of subgoals, which Reynolds called 'Steering Behaviors'. Examples of steering behaviors include obstacle avoidance, seek, flee and path following.

Whilst we have implemented several steering behaviors into our simulation, the one of direct relevance to this paper is wonder, which aims to randomly steer the creature around the environment. With wonder, the steering force is constrained to the surface of a circle or sphere in front of the creature. By making small random displacements to the steering force, the creature can retain overall steering direction, yet make a slight adjustment to its bearing. By varying the displacement size and rate, the characteristics of the creature's movement can be changed. For example small frequent changes in heading leads to smooth, deliberate arcs of movement reminiscent of a lizard, whilst large infrequent changes lead to a far more erratic, sporadic movement more akin to a house spider. The displacements can also be manually controlled, over-riding the random element of the wonder behavior.

Once the steering forces have been determined by the separate steering behaviors, they are combined into a single control signal and passed to a locomotion system which converts the force into the actual constrained motion of the creature's body. Based on a point mass approximation, forward Euler integration is used by the locomotion layer to update the creature's position. At each simulation step, the behaviorally determined steering force is applied to the creature's point mass. This produces an acceleration, which is added to the current velocity of the creature to produce a new velocity. Finally, this new velocity is added to the creature's old position.

There are two main advantages of using this technique. Firstly with an appropriate convention for communicating control signs, steering behaviors can be completely independent of the specific locomotion scheme. Whilst our implementation is arachnid based, this independence allows for different locomotion schemes to be developed without requiring an overhaul of other layers. The second advantage is that the locomotion of the creature can be dependant, or independent from its animated depiction. This allows us to couple a simple locomotion model with an adaptive animation model, the resulting hybrid having the flexibility and simplicity of a point mass system, yet retaining the more realistic dynamic animation of a physically simulated model.

3.2. Changes in orientation

An observation made by the primary author is that the orientation of a creature capable of scaling walls and walking across ceilings is always roughly parallel to the surface it is currently traversing. By coupling this observation with the point mass approximation based locomotion system, our simulation is able to update the position and orientation of the creature's body as it walks on an arbitrary surface. A map of the environment is generated at initialization, which stores the edge connectivity of each polygon. The creature keeps track of which polygon it is currently standing on using a polygon-ray intersection test, the ray being cast from the creature's origin in the direction of its negative up vector. The test is carried out every set number of simulation steps, against the polygons immediately surrounding the last known current polygon, taken from the aforementioned connectivity map.

One thing to take into consideration is the frequency of ray-polygon intersection tests compared to the velocity of the creature and the resolution of the environment. If the creature moves outside of the immediate polygon neighborhood between tests, the test will need to incrementally widen its search in surrounding neighborhoods for the new intersected polygon – effectively forfeiting any advantages of infrequent tests.

Once the intersected polygon is located, the up vector of the creature's orientation needs to be set to the negative of the polygon's normal, updating the creature's bearing. Simply overwriting the creature's up vector when it reaches a new polygon leads to erratic and unrealistic changes in orientation. For smoother transitions, the system interpolates between the old and new orientation over a number of simulation steps either linearly or using a more fluid cosine interpolation or a slow in slow out curve. If the creature reaches a new polygon before the current interpolation can finish, the process resets and the system interpolates between the current orientation of the creature and the new target orientation.

This technique is adequate for environments which are suitably convex or concave. For example, the creature will smoothly and easily traverse the interior or exterior of a spherical object. Similarly, the creature will follow the surface detail of an arbitrary height-map generated terrain. However, the technique is unsuitable for environments with irregular, angular or chaotic surface features such as shear cliffs, fissures and overhangs. In order for the creature to be able to successfully navigate such features, the locomotion/orientation engine needs to be extended. The problem lies in extreme changes in angle between polygons, specifically angles approaching 90 degrees and larger. Consider a creature walking along the surface of a cube. Using the above technique only, the creature will successfully traverse across the surface of any of the cube's sides, but at an edge, the polygon-ray test will fail as the ray does not intersect the immediate polygon on the subsequent side.

Our current solution to this problem is to instigate a scanning mechanism if the ray projected from the creature's origin fails to intersect any polygons. As the surface is continuous, such an event will only occur when the creature has reached a right angled or overhanging edge. When the creature finds itself in this position, instead of projecting the ray along its negative up vector, it incrementally scans it back and forth along its forward vector axis at increasing angles. This is somewhat akin to simplified rhythmic searching movements in arthropods when ground contact is absent [Dur01]. Once the polygon has been intersected by the scanning ray, the creature can re-orientate itself along this new plane and revert back to the standard test.

4. A Spider Model

The physical spider model created for the system comprises a skeletal model deforming a polygonal mesh. A backbone structure links the head, abdomen and thorax sections of the spider's body, with a further four bones extending from the head for optional animation of the chelicera and palp appendages.

Each of the eight skeletal limbs attach to the abdomen, and are composed of five bones connected via joints, with a further parent joint attaching the leg to the spider's body.

Whilst this model is not entirely anatomically correct, the detail is adequate for realistic animation. For example, two of the joints present in a real spider leg have been merged into a single, more dexterous joint, as the benefits of simulating both joints individually are negligible.

The prototype model is based on a Huntsman spider, although any hypothetical arachnid can be simulated by adjusting the bone length and joint restrictions. Furthermore, the system can be easily modified to simulate other arthropods and with some further extensions, lizards.

4.1. Generating motion

In order to generate the dynamic walking animation of the spider, a procedural model is combined with an inverse kinematics solver.

Each leg has an associated target point, which represents the desired position of the tip or foot at that particular point in the animation. The procedural model describes the path through which the target, and therefore the leg tip, travels during a single stride. The source of the stride path generated by the procedural model is a single or group of simple mathematical equations, an example of which can be seen in figure 1. The number of separate equations used is dependant on the required complexity or subtlety of the animation. For example, varying stride types can be linked with the behavioral states of the creature, and interpolation allows for smooth transitions between different gaits as the creature changes its speed, orientation or intent.

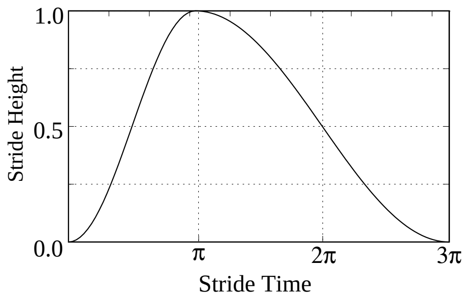

Figure 1: *Example step path.*

The joint rotations required to achieve a limb configuration which satisfies the goal of end-effector/target intersection is governed by a differential-based inverse kinematics system [WW92], [ZB94]. In order for the inverse kinematics solver to produce natural-looking postures, suitable constraints based on the simplified anatomical arrangement of a spider are modeled into the skeletal system.

A basic leg model consists of a body attaching parent joint capable of three degrees of freedom (ball-socket joint), and five inter-bone joints each with one degree of freedom (vertical hinge joints). These joint orientation restrictions are designed to reduce the complexity of the inverse kinematic configuration space, yet retain the minimum limb flexibility required to reproduce realistic arachnid animation. For a more accurate model, extra dexterity is added to individual inter-bone joints in the form of a restricted pivot, allowing for limited lateral movement along the leg.

Comfortable joint configurations are also encouraged, by adding springs to the joints. Each joint has a comfortable range of movement where there is no resistance to rotation. Increased deviations beyond the extremities, described by a maximum and minimum comfort angle, result in increased resistance until the joint reaches an absolute limit, after which no further rotation is possible. These individual rotational limits are implemented via Spherical Interpolation (SLERP). Joints with one degree of freedom have two corresponding 'min' and 'max' quaternions, the limits being enforced by interpolating between them. For joints with higher degrees of freedom, a rectangular SLERP is used where the rotational limits are described by four quaternions which define a hemisphere of possible movement.

The gait timing is controlled by a state machine, which cycles through the legs based on a variety of octopod gaits [Par98] [KLSK01]. In a tetrapod walking gait, legs on opposite sides work together as 'tetrapods', and the four legs in each group move in a wave-like succession, approximately 180 degrees out of phase with each other. Figure 2 shows an alternating tetrapod walking gait, which we deem to be the gait that produces the most realistic arachnid animation in our simulation. When the state machine triggers a leg to move, a targeting system calculates the end position of its associated target point by projecting by a stride size magnitude from the leg tip along the creature's forward vector. Once the target end point has been found, the procedural model will generate the path through which the target will travel during the stride. This path is rotated so that it is aligned along the same plane at the target's start and projected end point.

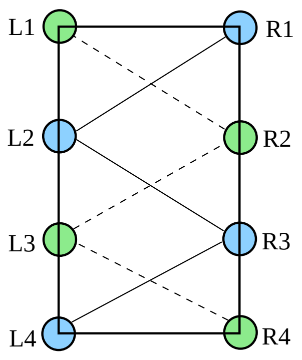

Figure 2: *Typical arachnid Alternating Tetrapod Gait. .*

The targeting system uses an extension of the alignment system used to orientate the creature's body. The end target point is approximated via projection, and then the exact position of the target on the landscape is found by calculating the point where a ray cast along both the creature's positive and negative up vector intersects the environment. It is necessary to cast the ray in both directions as the approximated end point location might be underneath the landscape, due to elevation changes in the surface immediately in front of the creature.

When a leg has reached the end point of the stride path, it is anchored at that target end point until its next stride is triggered by the gait generator. In the meantime, the inverse kinematics solver pivots the leg around its anchored point as the spider's body continues to move.

The joint linking the abdomen and thorax bones is influenced by the difference in elevation between the front and back four legs. The larger the average difference in elevation, the more bend at the joint. This is particularly effective when the spider is undergoing a large orientation change, such as crawling over a steep edge.

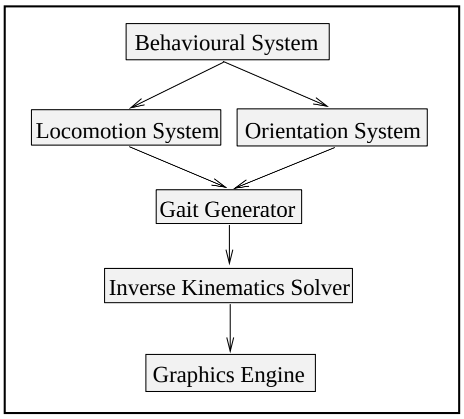

Figure 3: *Relationships between simulation components.*

5. A Spider implementation

In our current implementation, the system can produce realtime simulation of a spider walking across an uneven surface, such as that produced by a heightmap. The spider can also navigate the surface of both convex and concave objects.

5.1. Orientation and Targeting issues

However, dynamic movement over objects with angular and chaotic features are limited to controlled situations. The inadequacy lies in the method used by the limb targeting system in calculating the end point of a stride path, and the similar technique used to align the creature's body with the surface. The scanning approach used by the orientation system for polygon angles towards the perpendicular works perfectly when the creature reaches the edge of a cliff, but is not suitable for orientation changes required to scale walls. For example, the spider is capable of correctly re-orientating itself when moving across the exterior surface of a cube, but not the interior.

The method is also unsuitable for the targeting system, which further suffers in accuracy due to the fact that the approximation used to find the end of stride position does not take into consideration the topography of the surface between initial and end positions.

Our immediate research is concerned with developing a terrain alignment system which is robust enough for any surface, and an accurate targeting system for calculating end of stride positions on the surface. For the latter, one possible approach currently under investigation is to project a stride length ray along the surface of the environment in the forward direction of the creature, clipping and re-projecting the ray at any overhangs/intersections. Such a method would not only accurately predict the end position of a stride, but also take into consideration the elevation of the terrain in the process.

A similar technique could be used for aligning the creature to the surface, with the projection acting as a probe sensing immediate surface changes directly in the creature's path. Such a system would solve the problem of scaling walls, as the right angled polygon will be sensed prior to the creature's arrival.

5.2. Inverse Kinematics Solver optimization

Our differential based inverse kinematics solver produces realistic articulation for each of the spider's eight limbs as they stride and pivot around the body. However, calculating the configuration of eight limbs is computationally expensive, and therefore we are actively seeking to optimize this part of the simulation. Our plan is to convert our inverse kinematics solver so that it uses a half-jacobian approach [MM04], which takes advantage of the fact that a spider effectively walks on the tip of its legs - the orientation of the tip being immaterial. This conversion will yield a considerable reduction in computational cost, increasing the number of creatures we can simultaneously model in an environment.

5.3. Realism and behavior enhancement

This work is of a preliminary nature. However, we are currently performing user studies to evaluate the realism of the animation generated by our simulation.

The inclusion of a behavior which causes the spider to stop suddenly and twitch its palp appendages, before continuing to move after a random amount of time received very positive feedback, especially from arachnophobes. The unpredictability factor given by this relatively simple behavior further helps increase the suspension of disbelief.

We plan to extend the creature's behavioral system, in order to further enhance the realism of the simulation. Possible behaviors under consideration include the slowing down when large orientation changes have been sensed, preying and evasion. Wall following is also a behavior we seek to implement, which will form a decision making process of when to move parallel to a wall and when to climb up.

The implementation of surface properties is also under consideration. This addition would make certain environmental surfaces, such as glass, difficult or impossible to climb.

6. References

References

[Bow77] BOWERMAN R.: The control of arthropod walking. *Comp Biochem Physiol 56* (1977), 231–247.

[Bow81] BOWERMAN R.: Arachnid locomotion. *Locomotion and Energetics in Arthropods* (1981), 73–102.

[BQCR97] BEER R. D., QUINN R. D., CHIEL H. J., RITZMANN R. E.: Biologically inspired approaches to robotics: what can we learn from insects? *Commun. ACM 40*, 3 (1997), 30–38.

[BW97] BARRUS J. W., WATERS R. C.: Qota: a fast, multi-purpose algorithm for terrain following in virtual environments. In *VRML '97: Proceedings of the second symposium on Virtual reality modeling language* (1997), ACM Press, pp. 59–ff.

[CDK*99] CRUSE H., DEAN J., KINDERMANN T., SCHMITZ J., SCHUMM M.: Walknet - a decentralized architecture for the control of walking behavior based on insect studies. *Hybrid Information Processing in Adaptive Autonomous Vehicles* (1999).

[Dur01] DURR V.: Stereotypic leg searching movements in the stick insect: kinematic analysis, behavioural context and simulation. In *Journal of Experimental Biology, Vol 204, Issue 9* (2001), pp. 1589–1604.

[Fer95] FERRELL C.: A comparison of three insect-inspired locomotion controllers. *Robotics and Autonomous Systems 16* (1995), 135–159.

[KLSK01] KLAASSEN B., LINNEMANN R., SPENNEBERG D., KIRCHNER F.: Biomimetic walking robot scorpion: Control and modelling. In *Proceedings of the 9th International Symposium on Intelligent Robotic Systems, pages 101-108* (2001).

[MFCD99] MULTON F., FRANCE L., CANI M.-P., DEBUNNE G.: Computer animation of human walking: a survey. *The Journal of Visualization and Computer Animation 10* (1999), 39–54.

[MM04] M. MEREDITH S. M.: Using a half-jacobian for real-time inverse kinematics. In *International Conference on Computer Games: Artificial Intelligence, Design and Education* (November 2004).

[MZ90] MCKENNA M., ZELTZER D.: Dynamic simulation of autonomous legged locomotion. *Computer Graphics 24*, 4 (August 1990).

[Par98] PARKER G.: Generating arachnid robot gaits with cyclic genetic algorithms. In *Genetic Programming 1998: Proceedings of the Third Annual Conference* (1998), pp. 576–583.

[Rey99] REYNOLDS C.: Steering behaviors for autonomous characters. In *Game Developers Conference 1999* (1999).

[SM01] SUN H. C., METAXAS D. N.: Automating gait generation. In *SIGGRAPH* (2001), pp. 261–270.

[Ste97] STEED A.: Efficient navigation around complex virtual environments. In *VRST '97: Proceedings of the ACM symposium on Virtual reality software and technology* (1997), ACM Press, pp. 173–180.

[WW92] WATT A., WATT M.: *Advanced animation and rendering techniques*. Addison-Wesley, 1992.

[ZB94] ZHAO J., BADLER N. I.: Inverse kinematics positioning using nonlinear programming for highly articulated figures. *ACM Transactions on Graphics 13*, 4 (1994), 313–336.

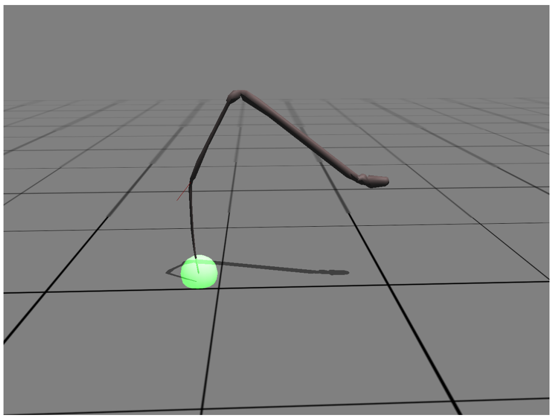

Figure 4: *Single insect limb with Inverse Kinematic Solver.*

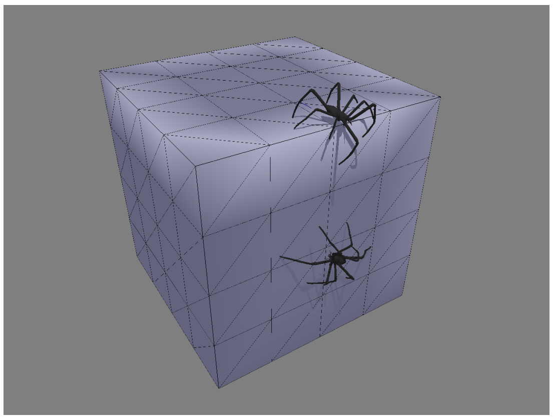

Figure 7: *Time-lapse view of spider walking on a cube.*

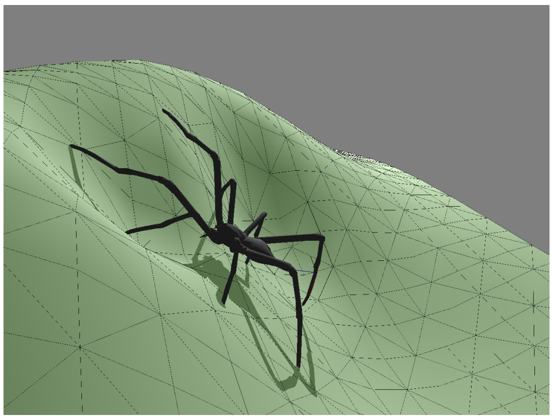

Figure 5: *Spider climbing a hill.*

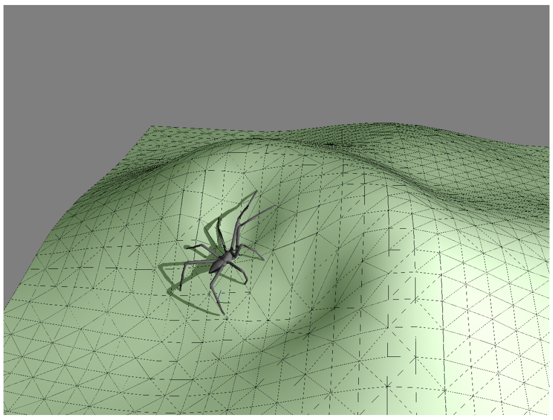

Figure 8: *Alternative view of Figure 5.*

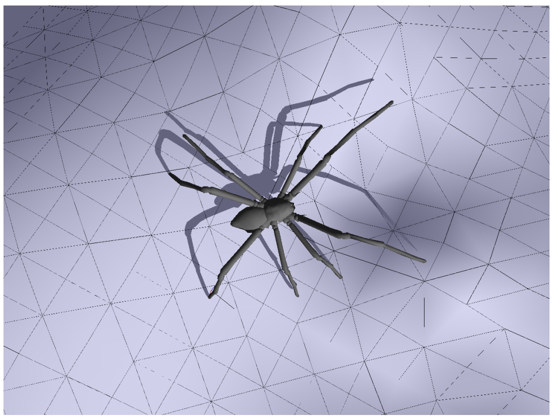

Figure 6: *Aerial view of spider walking across an uneven terrain.*

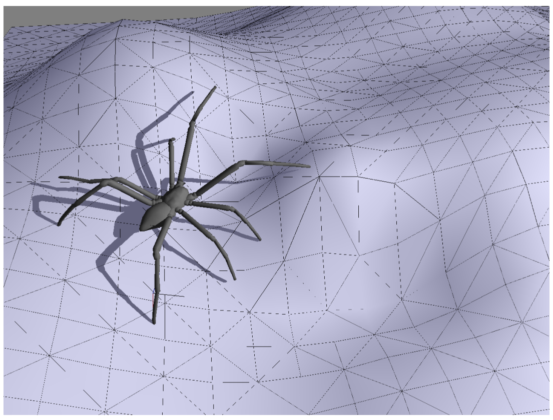

Figure 9: *Alternative view of Figure 6.*