Issues in the Physics of a Motocross Simulation

Benoit Chaperot, Colin Fyfe,
School of Computing,
The University of Paisley, Paisley, PA1 2BE, SCOTLAND.
email: benoit.chaperot, colin.fyfe@paisley.ac.uk

Abstract

In this paper, we investigate the use of a rigid body simulation engine to simulate a motocross bike. We discuss the movement of bikes from a theoretical perspective and apply this perspective to a simple motocross bike. This initial model suffers from control problems and we introduce several extensions which lead to a stable and enjoyable game. We believe that we have created the first realistic motocross simulator (as opposed to arcade game).

1 Introduction

Rigid body simulation, or physics simulation, is a method for simulating mechanical systems. It is generally present as a piece of software (library), used as part of another piece of software (in this case a video game).

Motocross Madness is a motocross game, released in year 2000; it is a popular and fun game to play. It does not make use of rigid body simulation. Other games, like MX Unleashed have been released since (see [10] and [9]), are nearly equally fun to play, and make use of rigid body simulation. However the simulation is used not to make the control of the bike more realistic, but to make the animation more attractive to the eye. One can see the rider moving on the bike, and the suspensions working as the bike goes over bumps, but the bike handles in a unrealistic way. These are called arcade games, as opposed to simulator games. In arcade games, fun and game play is preferred to simulation and realism.

The main difference between an arcade racing game and a simulation racing game is that in an arcade racing game the behavior of the vehicle is controlled by a set of rules, procedures and animations, whereas for a simulation the behavior of the vehicles is controlled by the physical and mechanical properties of the vehicle, and by the physics engine.

Simulators can offer a different, still enjoyable gaming experience, as proved by the success of games like Gran Tourismo. The use of rigid body simulation for vehicle simulation is not new. The library used for vehicle simulation can either be:

- A general purposes rigid body simulation library (for example [6], [11], [13], [12] or [4]); this solution has the advantage that it gives a lot of freedom and flexibility to developers to experiment and implement realistic vehicle simulation.
- A vehicle simulation library ([5] or [1]); this works well since the library is dedicated to vehicle simulation; it makes development easier; however, this solution may not offer as much flexibility as when using a general purposes rigid body simulation library.
- In house solution. This offers the most flexibility; however it may be very costly in terms of development.

There is no motorbike simulator on the market; this, together with a strong interest in bikes, simulators and video games, is our main motivation for creating what can be considered as the first motocross simulator.

In this paper, we investigate the use of rigid body simulation, to simulate a motocross bike, in the context of the Motocross The Force game. Motocross The Force is a motocross game featuring terrain rendering and rigid body simulation. An example of it in use can be seen at

`http://cis.paisley.ac.uk/chap-ci1`

The game has been developed and is still being developed in conjunction with Eric Breistroffer (2D and 3D artist). First, we have an overview of the theory behind bike riding, then we detail our approach to using rigid body simulation to simulate a motorbike and some of the improvements made over the original models. Finally we conclude by discussing the right trade off between simulation and arcade methods.

2 Riding a bike: theory

First, it is worth noting that a motorbike is by nature stable; this means that, in normal conditions, while riding a motorbike, the handle bars and front fork do not oscillate; the bike tends not to lean to one side, and it tends to go in a straight line.

There are two main things that make bikes naturally stable:

- Gyroscopic precession: this is a phenomenon occurring in rotating bodies in which an applied force is manifest 90 degrees later in the direction of rotation from where the force was applied.
- Rake angle and trail: as described on The Master Strategy Group site [2], wobble and weave are diminished because, when the wheel is pointing at an angle other than straight ahead, the contact patch is not in alignment with the direction of travel of the

bike, that is, a slip angle is created. A restoring force is applied to the contact patch by the ground which attempts to force that alignment. Thus, because of trail, the front wheel tries to go in a straight line (see Figure 1).

In practice, a rider can ride a motorbike with no hands, except to operate the throttle.

Let's now consider the main forces acting on the motorbike; for simplicity it is assumed that the rider is attached to the bike, and bike and rider can be considered as one body. The three main forces acting on the bike are:

- The weight, acting down: $F_g = mg$ with **m** the total mass, and **g** the gravity.
- The centrifugal force, acting horizontally, directed towards the outside of the turn:

$$F_c = \frac{mv^2}{r} \tag{1}$$

  with **m** the total mass, **v** the linear velocity, and **r** the turn radius.
- The contact force, from the contact between the tyres and the ground.

The sum of these three forces can be assumed to be zero and the triangle of forces closed, if the bike is balanced.

Other forces, including traction, inertia, air friction, gyroscopic precession are also acting on the bike, but these can be ignored for now for clarity, mainly because these forces have no effect or no negative effect on the balance of the bike.

From Figure 1, and with the concept of these three main forces acting on the motorbike, one can assume that what makes a motorbike turn, is not the direct action on the handle bar; instead, it is the angle the bike is making with the ground (roll angle).

Besides, one can notice that the only force that can be controlled by the rider is the horizontal component of the ground contact force. This horizontal force, which acts at ground level, determines the roll angle the bike is making with the ground and is obtained by action on the handle bars.

As an example, let say the bike is going in a straight line; the rider wants to turn right:

1. The rider would first counter steer left, in order to pull the front wheel contact patch to the left, and put the weight of the bike to the right of the contact patches.
2. The weight makes the bike lean to the right.
3. The rider would then gently steer right, to close the triangle of forces, and balance the bike; the bike can be assumed to be balanced when the sum of weight and centrifugal forces lie in between the two contact patches.

The procedure is reversed if the rider wants to turn left or wants to go back to a straight line.

We are confident that most of the mechanical phenomena described here can be reproduced using physics simulation.

3 Physics Simulation model

Open Dynamics Engine (ODE) [6] was chosen for the simulation, because it is an open source project, with a large community of users maintaining it and enhancing it. Also it is probably as fast, accurate and stable as expensive commercial rigid body simulation packages. Open source gives the possibility to modify the engine and add the features that are missing for a particular requirement. LUA is a script engine which is used for scene creation. It allows modifying the simulation scene without having to recompile the executable. It allows the separation of code and scene data.

3.1 Collision Model

On a car simulation, wheels are often modelled as spheres. This is not appropriate in the case of our motorbike simulation, because as the bike makes an angle with the ground, the effective radius of the bike would reduce. Other primitives have been tested, such as thin cylinder primitives (disks), but these primitives proved not to be very stable. Another problem with thin disks is that because of the limited time step used for the simulation (1/100 of a second), a thin disk can fall through a plane, if lying against this plane. With this limited time step the simulation engine does not handle collision between primitives with limited thickness well (due to gravity, a very thin object can travel more than its thickness during one time step, and the collision with a coplanar plane can be missed).

For the rendering, a terrain engine is created. All vertices are positioned on a regular grid (horizontal x and y coordinates), but with different height (z coordinate). This makes the creation and editing of terrains easy and efficient because all the terrain can be created, edited, and stored in memory as a two dimensional height map. A triangle mesh collision primitive could have been used, but it would not take advantage of the two dimensional distribution of vertices, hence would not be as efficient as a collision primitive that will take advantage of this particular distribution.

For these reasons, two collision primitives have been created for ODE. One primitive is a cone, with high radius to length ratio, which is used for the wheels. The other primitive is a terrain, making full use of the particular distribution of vertices.

One problem with creating new primitives is that for each primitive, a call back function is needed for collision with any other primitive. Hence the number of collision

functions grows exponentially with the number of primitives. For the terrain the problem has been solved by considering the collision of primitives with the terrain as collisions of primitives with the planes and edges making the terrain; hence reducing the collision problem as reusing all the other primitives collision functions with planes and rays. For the cone, only two collision functions have been implemented, collision between cone and plane, and collision between cone and ray, in order for cones to collide with terrains. For any other collision, the sphere collision functions are used; this proved not to be a problem and any discrepancies are not noticeable in the vast majority of cases.

3.2 Dynamic Model

Dynamic bodies are used for the simulation. Each dynamic body can be associated with one or more collision objects. More than one collision object can be used on the same body to create a simulation object with a complex collision shape. Rigid bodies have mass and inertia tensors; these masses and inertia tensors can be set independently from the shape of the associated collision objects. As an example, wheels are set to have cones as collision object, but have hollow cylinders for mass and inertia tensor. As a first attempt, a fork and front wheel have been modelled using ODE joints. The wheel was attached to the fork using a Hinge joint, and the fork is attached to the bike frame using a Hinge2 (suspension and double rotation joint). This proved not to be successful; because of the limited time step and corresponding lack of accuracy of the physics engine, each joint introduces an error, and the sum of the two errors means the front wheel looked as if it was very loose. Instead, a Hinge2 joint has been used to attach the front wheel directly to the frame; this proved to be more successful but removes the trail as discussed above. The fork is attached to the frame, using a Hinge joint and is given the same rotation as the front wheel Hinge2 top axis. The rear wheel is also attached to the frame using a Hinge2 joint. The rider's trunk is attached to the frame using a Fixed joint, to prevent him from falling off the bike. All riders' articulations, elbows, knees, wrists, are modelled using Universal, Hinge, or ball and socket joints, with limits set on the joints to prevent the rider doing forbidden moves, and allow him to be animated by the simulation in a realistic way. All simulation objects' size, position and orientations, and joints' position and orientation are obtained procedurally, from the rendering meshes. The user has the possibility, through LUA script, to choose the collision primitive for each object, set joint types, and modify objects' sizes, positions and orientations, and joint positions and orientations, masses and inertia tensors.

An AMotor (angular motor) is used on the rear wheel, to allow for bike acceleration and braking.

All masses, inertia tensors, and mass parameters are set experimentally.

The player controls are the direct action on the handle bar, and the torque applied on the rear wheel.

3.3 Initial Results

All tests and experiments have been carried out using a Evaluation Panel composed of three regular gamers; these regular gamers evaluated the game at every stage of development and fed back to the main game developer criticisms and comments. To ride a bike, the user has 2 pairs of controls used to turn left/right or accelerate/decelerate the bike. The bike is extremely difficult to control.

1. It keeps on falling onto its sides; it is nearly impossible to control the balance of the bike by direct action on the handle bar.

2. It keeps on flipping while accelerating or decelerating; it is difficult to find the right angular motor parameters (target angular velocity and torque to achieve acceleration and deceleration).

3. The biker seems very rigid, because the rider is glued to the bike seat.

4 Improving the simulation

A few improvements to the original models have been made in order to improve the simulation.

4.1 Indirect action on the handle bar

What the player really wants is to turn left or right; turning left and right is not achieved by direct action on the handlebar; instead, as described above, it is the angle the bike is making with the ground, that makes the bike turn, and this angle is obtained through action on the handlebar. Hence, as an experiment, let us interpret the left/right player control of the bike as a target roll angle the bike is to make with the ground; and let the game engine evaluate the appropriate action to apply on the handle bar in order to achieve this target angle. As a first consideration, we can state that the action on the handle bar is dependant on the bike's velocity; a rider does not turn the handle bar as much while riding at high speed than while riding at low speed. We also state that the action of the handle bar is dependant on the difference between the current bike roll angle and the target bike roll angle, and also dependant on the current bike roll angle. As an experiment, we test with the following rotation for the handle bar:

$$R_h = \frac{C_s * (A_c - A_t) + C_n * A_c}{v} \tag{2}$$

With $\mathbf{A}_c$, the current bike roll angle, $\mathbf{A}_t$ the target bike roll angle (mapped as left/right control of player), $\mathbf{v}$ the linear velocity, and $\mathbf{C}_s$ and $\mathbf{C}_n$ two parameters to be determined experimentally.

After a few tests to find appropriate values for $\mathbf{C}_s$ and $\mathbf{C}_n$, this proved very successful; the bike does not fall onto its sides anymore and balance is maintained. However, it is still difficult for the player to fully control the direction of the bike.

4.2 Adding a force

As an experiment, instead of using an angular motor for the rear wheel, a force is applied directly on the bike frame. Applying directly a force is appropriate because it conveys the feeling of a continuous thrust one can have while riding a motocross bike. This feeling of continuous thrust is in practice mainly due to the loose traction between the bike wheels and the ground.

This proved to be very successful; the bike was not flipping anymore at acceleration and deceleration. Beside, by changing the position where this force is applied, it is possible to get the front wheel to rise while accelerating hard, and the rear wheel to rise while decelerating hard.

4.3 Adding torques

To make the bike even more stable, as an experiment, the bike frame is attached to the static environment using an angular motor. For the three bike axes, target angles are set for the frame in relation to the static environment (ground), and the angular motor applies torques to the frame in order to achieve those target angles. Spring and damping can also be adjusted on this joint, in order to obtain the right joint behavior. The ODE AMotor joint had to be modified, in order to allow an object to be attached to the static environment, and to accept angles outside the $\{-\pi, \pi\}$ range. This proved extremely successful; the bike was a lot more stable, and it was easier to turn. This also allowed for one extra control, lean forward or backward (biker weight on the front or on the rear of the bike). More details about AMotors can be found in the ODE use guide [8]:

4.4 Detaching the rider from the motorbike

A new joint called linear motor, has been implemented. It is very similar to the angular motor, but works with forces and translations instead of torques and rotations. The rider trunk is attached to the bike frame using this joint, and the fixed joint is removed. This proved very successful. The rider with his body weight is now able to absorb part of the shocks, just as a real rider would. The simulation looks more realistic, and the bike is more stable. More information about creating new joints in ODE can be found in this paper [7].

4.5 Simulating trail

As seen above, because a Hinge2 joint has been used, there is no more trail to force the front wheel in alignment with the ground. A trail can be simulated by forcing the front wheel in the direction of the moving ground. The rotation for the handle bar now becomes:

$$R_h = \frac{C_s * (A_c - A_t) + C_n * A_c + C_y * A_y}{v} \quad (3)$$

With $\mathbf{A}_c$, the current bike roll angle, $\mathbf{A}_t$ the target bike roll angle (mapped as left/right control of player), $\mathbf{A}_y$ the current bike yawl angle, or difference angle between the forward direction of the bike frame, and the velocity vector of the bike. $\mathbf{v}$ is the linear velocity. $\mathbf{C}_s$, $\mathbf{C}_n$ and $\mathbf{C}_y$ are three parameters to be determined experimentally. This also proved to be successful as it made the bike even more stable.

5 Conclusion

Simulating a motorbike is more difficult than simulating a car, because a lot more bodies, joints and mechanical phenomena are involved. We believe that this is the first time such a motocross simulator has been successfully created.

Simple simulation models can make a bike realistic but makes the game totally unplayable. Modifying the models, to make the game playable may involve introducing controls and objects which are unrealistic. The work is not finished yet, but so far the simulated bike seems realistic, and the game seems fairly easy and fun to play.

The game is slightly harder to play than most arcade motocross games, but making the game easier would mean adding more unrealistic controls, cutting on the simulation, and the game would loose some of its appeal. It is also a choice to have the game appeal to a large public, and not only to young children. Current and future work involve fine tuning the simulation, and creating more bikes.

We [3] have also trained artificial neural networks using backpropagation and evolutionary algorithms to learn to ride such bikes. Future work will also investigate whether other computational intelligence techniques can be used for this purpose.

References

[1] http://www.carsim.com/. Technical report, Mechanical Simulation, 2005.

[2] C. Anthony and J. Davis. http://www.msgroup.org. Technical report, The Master Strategy Group, 2005.

[3] B. Chaperot and C. Fyfe. Motocross and artificial neural networks. In *Game Design And Technology Workshop 2005*. TBA, 2005.

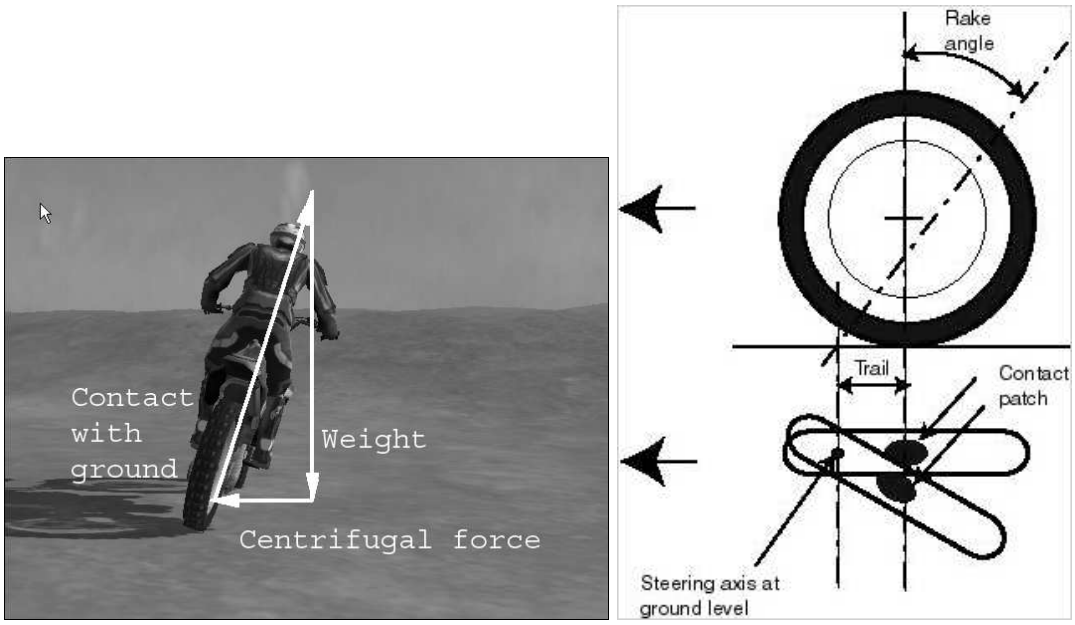

Figure 1: The three main forces acting on the bike. Rake angle and trail are used to make the bike stable, figure used with permission of The Master Strategy Group http://www.msgroup.org

Figure 2: The bike has seen in the game and its associated collision objects used for the simulation

[4] David Lam. http://www.tokamakphysics.com/. Technical report, 2005.

[5] E. Laptev. http://www.oxforddynamics.co.uk/. Technical report, Oxforddynamics, 2005.

[6] R. Smith. http://www.ode.org/. Technical report, 2005.

[7] R. Smith. http://www.ode.org/joints.pdf. Technical report, 2005.

[8] R. Smith. http://www.ode.org/ode-latest-userguide.html. Technical report, 2005.

[9] Various. http://ps2.ign.com/articles/445/445638p1.html. Technical report, IGN, 2003.

[10] Various. http://thq.com/game.asp?1052—46045. Technical report, Rainbow Studios, 2003.

[11] Various. http://www.havok.com/. Technical report, Havok, 2005.

[12] Various. http://www.novodex.com/. Technical report, AGEIA Technologies, 2005.

[13] Various. http://www.renderware.com/physics.asp/. Technical report, Criterion, 2005.