



RUMBLE ROCKETS

FUNCTIONAL SPECIFICATION

David Burkett – dburkett@purdue.edu – (260)450-6301

| | |
|-----------------------|---|
| Executive Summary | 1 |
| Menu System | 2 |
| Rules of Play | 3 |
| Track Placement | 4 |
| Level Design Tool | 5 |
| Obstacles and Pickups | 5 |
| Flight Controls | 6 |
| Gameplay Flowchart | 7 |

EXECUTIVE SUMMARY

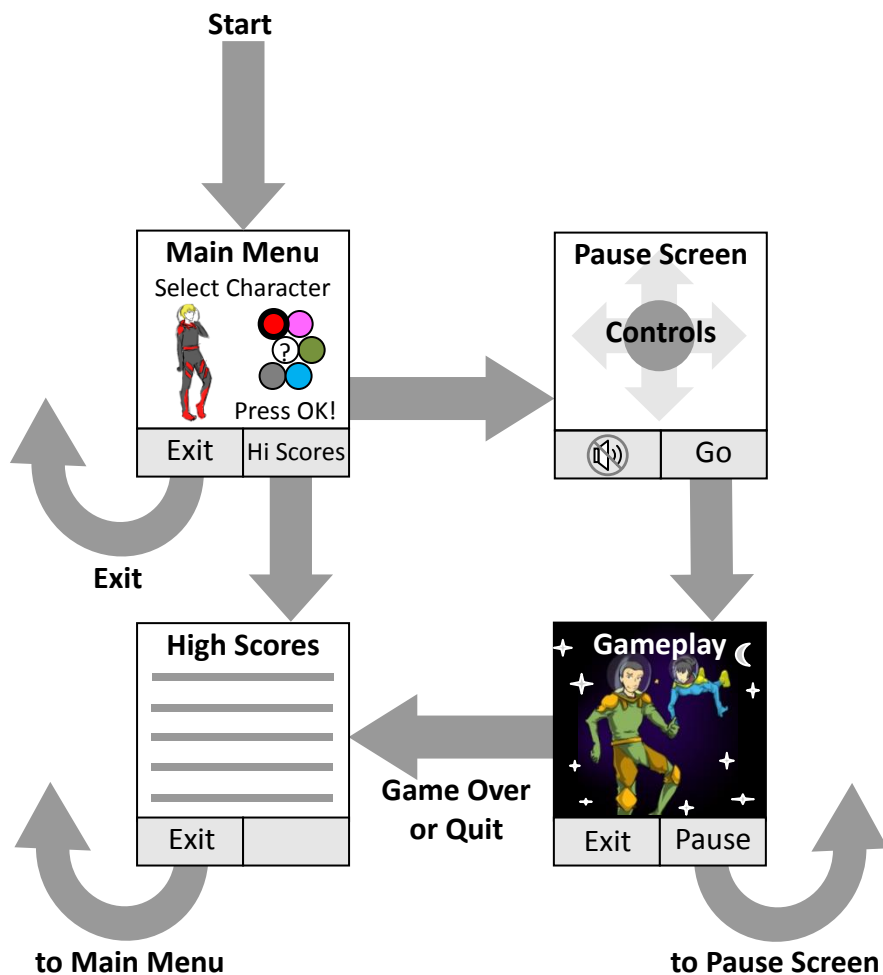
Starry skies and racetrack markers spin and whirl as you fly and tumble through beautiful deep space. The only thing between you and the vacuum is a winged and rumbling rocket-packed space suit. Your goal: to race your competitors to the next checkpoint and win!

Rumble Rockets is the new J2ME mobile game that defies the limits of the **rac**ing genre: immersed in **true 3D**, you race through twisting turns in all directions as rings of markers fly past on all sides describing the tube-like track space. Speed is controlled by a novel booster/break item system, and the track is new every time. Three things stand in the way: you must deftly avoid **obstacles** such as asteroids and the occasional stampeding herd of space cows. Also, you'll have to time the **catapult booster** just right at each race checkpoint. Finally, you must jockey against pushing and shoving **opponents** for the win.

But Rumble Rockets is a simple, fast-paced and fun game; it is not weighed down by complicated decisions or menu screen after menu screen. Just pick a racer and launch!

MENU SYSTEM

There is only one actual menu screen – character selection. On the Main Menu, pressing OK selects a character and proceeds to the Pause Screen to explain the controls. From here on, the right soft key toggles the pause. While paused, the left soft key toggles sound (muted by default). At all other times, the left soft key moves the player closer to the exit. After the game ends, the High Scores screen is displayed, and if they scored high enough, they can input their name.



RULES OF PLAY

Win State – to continue as long as possible and beat previous high scores.

Loss State – worse than 5th place at a checkpoint.

The Checkpoint – big high-tech rings that check the racer's rank every few minutes. When the racer passes inside, the checkpoint produces several beams that lock onto them. Like huge slingshot rubber bands, the beams stretch, bounce back and then catapult the racer forward. The boost is based on the racer's proximity to the center and their timing on the **Catapult Bar** (see below). If racers collide during this process, the first racer inside checkpoint will be unaffected and all others will take a normal collision; any "losers" already locked into the checkpoint will be ejected behind it.

Difficulty Ramp – each time the player passes through a checkpoint, the opponent AI improves, the track becomes more difficult, and the background scenery may change.

The Catapult Bar – While that is happening, the **Catapult Bar** appears. An indicator moves back and forth on this bar, marking where the **Catapult Button** is pressed (see **Flight Controls**). These marks decide how far the catapult stretches and how much boost the racer gets – the wider apart, the better. The indicator starts at the left, goes right, left, and then flies off to the right as the racer launches. If the first mark is in the left half, the second is in right half, or less than 2 marks are made, the boost is failed and the racer is released with a strong slow-down effect.

The Catapult Boost Equation – Give each point on the bar a value (0 to l from left to right). In this equation, v = default speed, m_1 and m_2 = the 2 mark positions, r = checkpoint radius, and d = racer's distance from center. The new speed is independent of the previous speed. Pre-compute the $\frac{4v}{r^2}$.

$$speed = m_1(l-m_2)(r-d)\frac{4v}{r^2}$$

Speed – Each racer has these values: **currentSpeed**, **idealSpeed**, and **position**. The ideal speed can never drop below the default speed. The only way for the player to affect their speed is through the boost/slow-down effects from racetrack objects and techniques - simply add those effects to the ideal speed. **AccelAdjust**, a global variable, is a standard number of game cycles it should take for each currentSpeed to reach idealSpeed. So to adjust each currentSpeed,

`currentSpeed += (idealSpeed-currentSpeed)/AccelAdjust;`

The idealSpeed may change each game cycle, and this adjustment incorporates that to better approximate a smooth curve.

Player Score – When a player pulls off a technique or grabs a power-up, add an appropriate value to the score. Also, every second or so, adjust the score like this:

`Score += (worstPossibleRank-currentRank)*smallConstant;`

At every checkpoint, make this adjustment:

`Score += (6-currentRank)*bigConstant + distanceAheadOfNextOpponent;`

Track Markers – appearing in rings of four dashed lines, the racer must not fly beyond these. Failing to fly through a ring of markers causes a significant slowdown effect, a haze over the screen, and an "Out Of Bounds" announcement until the racer reenters legal space. If they ever go backwards, "Wrong Way" will be announced. Markers are simply a flat texture with no model. Possible colors: **Regular**, **Turn Warning** (only one marker per ring), **Checkpoint** (all markers in the ring), **Backside** (any marker viewed from the outside).

Random Track Placement – most of the track will be randomly generated (see next section).

TRACK PLACEMENT

These path restrictions will govern the racetrack placement. To allow easy adjustment, use these **Global Variables**:

DefaultSpeed – the base speed for all racers.

MarkerRingDistance – a length such that the **Default Speed** equates to 2-3 marker rings/sec.

MaxAngle – the maximum angle of turn.

RotateRange – the angle a marker ring's rotation can deviate from the previous; the lower this value, the closer each rotation will be, and the smoother the track. Max 180°.

WarningAngle – minimum turn angle which requires advance warning.

WarningLookahead – the number of rings before a sharp turn that must have Warning Markers.

[ObjectType]Frequency – random multipliers for each type of object placed.

MaxObjectOffset – maximum distance from a ring center to a random object. Should be long enough that objects don't seem to appear only near the rings.

Marker Ring Variables:

center – the center point of a marker ring.

pointA – a point $\frac{1}{2}$ a Marker Ring Distance in front of a Ring Center, normal to the plane.

pointB – a point $\frac{1}{2}$ a Marker Ring Distance behind a Ring Center, normal to the plane.

rotation – the rotation of the **Inside Marker**.

color – the coloration of the markers. Regular, Warning (only on **Outside Marker**), or Checkered.

Riff Variables:

isRandomlyRotated – whether the riff is to be given a random rotation.

level – the number of checkpoints which must be reached before the riff is unlocked. Basic riffs are available initially (**level** = 0), while difficult riffs are unlocked later.

Placement Steps:

1) Random Marker Rings – Rotate the last ring placed, and place the new ring such that its **pointA** coincides with the previous **pointB** and forms no more than the **Max Angle** at that point. Pick 2 random values: x ($x \leq \text{MaxAngle}$) and y ($-\text{RotateRange} \leq y \leq \text{RotateRange}$). When placing the k^{th} ring,
`ring[k-1].rotation = ring[k-2].rotation + y;`
`ring[k] = new MarkerRing(ring[k-1], x);`

The track doubling onto itself must either have no significant effect or be impossible.

2) Random Objects – next, random obstacles and/or pickups are chosen with a random x , y and z offset from the ring center. Any objects placed outside the ring boundaries, will just be scenery.

3) Riffs – predefined “riffs” are randomly inserted several times per checkpoint distance. A “riff” is a set of ring and object positions stored in an array. Start the riff at the last ring placed. Riffs should be idiomatic and interesting in nature, such as a parking-garage-like rising spiral, a loop-de-loop with space-dust cloud bunkers lining the outer edge, a roller-coaster's first hill with boosters down the bottom middle, too-sharp turns with brakes pickups before them, corkscrews, etc. For level design inspiration, see roller-coasters, Mario Kart, F-Zero, etc. This is an opportunity to be very creative!

4) Checkpoints – a special riff that must occur every 1-2 minutes. It includes a checkpoint object between two straight-aways. The markers before the checkpoint should be checker-colored.

3) Marker Coloring – if the angle is sharp enough, color a number of the previous rings. Only one marker per ring (the **Outside Marker**) can be a Warning Marker. Do not affect Checkered markers.

LEVEL DESIGN TOOL

In order to easily design track riffs, a simple level design environment will be created in Maya or some other modeling package. Create a scene that has one of each of these objects:

Marker Ring – a grouped object consisting of a circle, a perpendicular line through the center of the circle (one **MarkerRingDistance** long), and 3 points (**center**, **pointA**, **pointB**). Add an arrow shape to **pointB** to differentiate it from **pointA**. If possible, make a skeleton for the object such that multiple rings can be joined together, **pointA** to **pointB**. Each ring's rotation should be based on the rotation of the joint at its **pointB**. Restrict the skeleton's movement based on the **MaxAngle** and the **RotateRange**. Markers may be added to the rings for artistic effect.

Generic Object – a steep pyramid which represents any object to be placed. The pyramid's pointy end defines the rotation of the object.

In order to use the level design environment, place the first **Marker Ring** at the origin of the scene with no rotation. Then duplicate these two objects and place them however you like, following the same path restrictions as random track placement. Finally, gather the position/rotation information for each **Marker Ring** and **Object** and store it all in an array to complete the riff.

TRACK OBJECTS

Track objects, such as obstacles, hazards and power-ups, are either randomly placed or located in riffs. Objects may only be for visual effect, or they may have one or more of these speed effects:

Collision – hit a physical object and bounce off, spinning out if the crash angle is sharp

Sticky (e.g. dust cloud) – slows the racer down while they pass through

Slippery – lose steering control and spin out, continuing on with the same velocity

Splat (e.g. alien creeps) – blinds the racer like a bug on the windshield

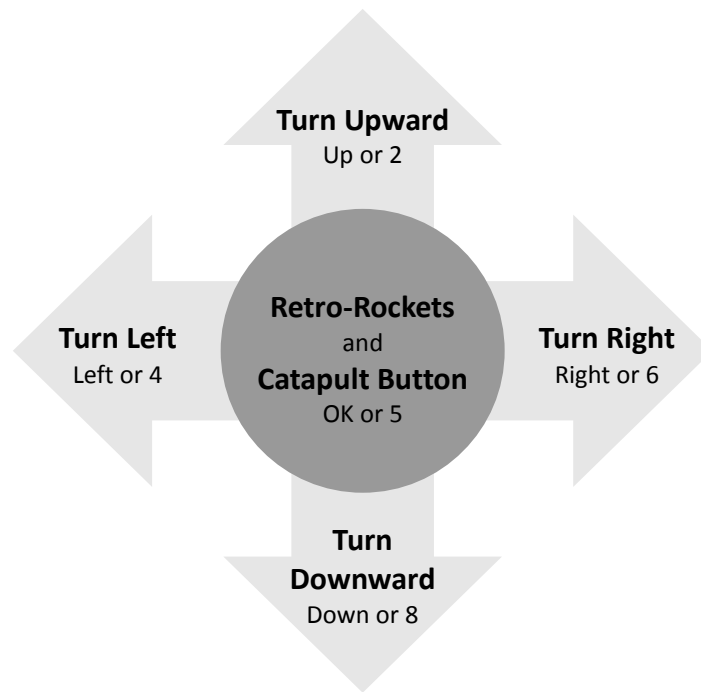
Crosswind (e.g. meteor shower) – steadily pushes the racer in one direction, which is chosen randomly when the object is placed.

Booster (Green Power-Up) – strong bonus boost effect. Like other boost effects, this turns the racer's rocket exhaust a hot blue-green for a moment.

Brakes (Red Power-Up) – strong bonus slow-down effect. Like other brakes effects, this turns the racer's rocket exhaust a duller red for a moment.

Unstoppable Speed (Gold Power-up) – the racer is temporarily invulnerable to all negative speed effects. The racer can only be slowed down by going off the track, the **Retro-Rocket Blast** or the **Power-Slide** techniques (see next section), while all boost effects continue to work normally. Any physical collision generates a "kapow" sprite (a cartoony explosion texture that grows and then disappears). Physical objects are destroyed and removed while hidden behind the kapow, and other racers are thrown to the side. The racer's rocket exhaust turns gold-colored.

FLIGHT CONTROLS



Techniques – Each successful technique will be followed by a splash announcement.

The **Slam** – just knock into an opponent, pushing them as well as possibly spinning them out.

The **Draft** – fly directly behind an opponent for about 2 seconds for a strong boost. Avoid the wash of their rockets or they'll turn it into a **Rocket-Exhaust Slam**, and don't hit them after the boost!

The **Rocket-Exhaust Slam** – hit an opponent with the wash of your rockets' exhaust for the most effective knockback and spin out, as well as a receiving a minor boost.

The **Retro-Rocket Blast** – firing this during straight flight causes a strong slow-down.

The **Power-Slide** – fire the **Retro-Rocket Blast** through a turn to pull sharply into it. The turn button does not need to be held anymore! Let off the retro-rockets to come out of it. The racer flies sideways in mid-turn and is moderately slowed. Instead of banking, the wings stay flat to the turn and only the retro-rocket on the inside of the turn fires. The racer will be seen from a side-view.

The **Barrel Roll** – while **Power-Sliding** in any direction, immediately press the turn button in the opposite direction (while still holding the retro-rockets). This technique causes a moderate boost, but causes the racer to temporarily lose all steering control. Good for timing offensive techniques, along with the **Retro-Rocket Blast**.

Racer Rotation and View – the racer rolls to bank through normal turns, returning to normal afterwards. The camera is in 3rd person and matches the racer's rotation; therefore the racer will be stationary on the screen (except for any idle/technique/other animation). The background inversely follows the racer's rotation, and translates based on the racer's turning.

GAMEPLAY FLOWCHART

