Mobile game take home - Planet Jumper

Objective

You are a game engineer who is pairing with a game designer to create a brand new game. The company you work at generally releases simple, small scope games, but you know that they have a history of repeatedly changing requirements when making new designs.

So when you are given the design for this next game, you know you must code everything with a mind towards inevitably changing it later by coding it in as robust a way as possible. But, you must also remember that other engineers will probably have to support this game after you've moved onto other things, so make sure the code is readable and easy to understand as well.

The Design

The designer says he has a really cool idea for a new game called "Planet Jumper," patent pending. In it, you control a little character who moves around on the surface of a very small planet and wants to get to a goal on top of another planet in the distance. He must jump in between the planets to reach his final goal. Along the way, he may encounter spikes that kill him when he touches them, and the planets may rotate.

Because this guy is a designer, he also gives you some pretty pictures to explain. Attached is "PlanetJumper - Design Doc.png" which is his guide for you. It comes with the following notes:
1. Guy begins at Start.
2. Player presses the <Right Arrow> key to move Guy clockwise on the surface of the planet.
3. Player presses the <Space Bar> key to make Guy jump. Once he passes the threshold* between two planets, he rotates the other way and lands on the other planet instead.
4. Player presses the <Left Arrow> key to move Guy counter-clockwise on the surface of the second planet.
5. Player presses the <Space Bar> key to make Guy jump. As he passes the threshold*, he flips around and lands on the third planet.
6. Player presses both the <Left Arrow> to begin moving Guy counter-clockwise. He also times it to press <Space Bar> to jump over the red spikes**. Because there is no planet near, he does not transfer to any other planets.
7. Guy lands in between two red spikes**.
8. Player repeats step 6 to jump over another red spike**.
9. Guy lands again, then the player uses <Left Arrow> to continue around the planet counter clockwise.
10. Player presses <Space Bar> to send guy past the threshold* to land on the last planet. Note that this time he doesn't go in a straight line. He takes off from planet 3 tangential to it, then once he passes the threshold* he drops onto the last planet tangential to that.
11. Player presses <Right Arrow> to move clockwise towards the exit.
12. Player reaches the exit and wins.

Clarifications

* The threshold is when the center of the player is closer to the surface of one planet than another.
** When the Guy hits a red spike, he gets zapped back to the start.
• The planets can rotate clockwise or counter-clockwise at any speed.
• The planets never move, their center is always the same.
• The start and finish never move.
• The Guy cannot fall off a planet or reach escape velocity. He *always* falls towards the tangent of whatever planet he is standing on.
• Planets are always perfect circles.

After delivering this doc to you, the designer goes on vacation! Do your best to create the game that was asked for given the information you have.

Delivery

The game should be built in JavaScript using HTML5 canvas, without any other intermediate tools or middleware. Please deliver a ZIP file that contains:
• An index.html file that launches your game.
• All source files in a sub-directory called "src"
• All assets in a sub-directory called "resources"
• A README.txt file that explains your solution and any extra features you decided to include, or any that you wanted

to include but could not (list the reasons why).

Lastly, we'd like to ask that you do not post this question or your solution online for others to use.

What We're Looking For

• Configurable - it should be easy to adjust level design, change the speed planets rotate, etc. The designer should be able to do this even though the designer can't code.
• Robustness - how easy would it be to expand more features onto your game, or make design changes?
• Readability
• Assume anything the designer has said could change, and plan for that in your implementation.
• Efficiency

Attached are images used for the concept art. You may use these, or anything else. You may include sounds, animations, whatever floats your schoo