Pangolin: Enhancement Proposal

Russell Dawes
Mohammed Gasmallah
Leonard Ha
Vincent Hung
Joseph Landy

Abstract

For this report we will be looking at the feasibility of adding a feature to SuperTuxKart. We will start by going over the two main methods we have derived in enhancing SuperTuxKart with this feature. We will then determine and analyze how adding this feature will affect the architecture of SuperTuxKart and a SAAM will be explained. The feature in particular is the addition of a new item that gives all players a warning before reversing the gravity for a short duration (~2-4 seconds) except for the player using the item.

Introduction and Overview

SuperTuxKart is known to be one of the best open source multi-platform games. Through it's game engine architecture, it is able to deliver on a fun and interesting game play independent of the operating system.

SuperTuxKart's gameplay is similar to Mario Kart but has changed greatly over time. It's mascots are from open source projects such as Mozilla Thunderbird, Beastie (BSD), Gnu (GNU), Hexley the Platypus (Darwin), Puffy (openBSD) and of course Tux (Linux). Currently the game has single player and local multiplayer, and they are planning a release of networked multiplayer.

With an addition of multiplayer, it can be expected that some competition will be desired, but not many of the items present in SuperTuxKart are varied in skill usage. There is only so much that can be done with a cake or a bowling ball for example. This is why we believe that an addition of an item that allows for a varying level of skill usage to be an enhancement of SuperTuxKart. This item is one that will give a warning to all players when used and will reverse gravity for 2-4 seconds. This allows for some competitive thinking and varied play. Since all momentum is carried and gravity is reversed, unskilled players will like this item because it allows them to catch up to and change the flow of the game for a short period of time. Skilled players will like this item as it allows them to plan shortcuts through bends and corners of the road. This gives a varied level of skill usage and variability to play which will ideally enhance competitive play.

There are risks in doing this as all elements of the item must be play tested and balance should be determined by listening to criticism. Timing of gravity reversal and severity of gravity reversal should be played around with, but ultimately the addition of the item should be rather gratifying. To compare and contrast the two methods to add this item, we have performed an SAAM to demonstrate the differences in the method.

Motivation

Currently, SuperTuxKart offers an immensely fun and satisfying casual play, but in order to better compete with others in the genre, SuperTuxKart is adapting network play. This leads to a new aspect of the genre which comes with multiplayer. Competition is necessary for many aspects of racing games and many players find this both a fun and necessary criteria for choosing to play a video game.

Many things have to be done before a game can go multiplayer, and one of the most important is making sure that the current gameplay is enough for the multiplayer to be a compelling/balanced gameplay. We find that SuperTuxKart almost reaches this goal, but to enhance the game, we believe that adding a low barrier to entry, but high skill cap item would enhance gameplay and balance for users.

This item would have to be one that interacts with many elements of the gameplay to provide a simple to understand yet interesting result. Allowing user that are new to the game to have fun and make an impact on the race, while also allowing a degree of balance so that other users can use this item to their advantage.

Method 1

The first method for implementing this item, would be to directly change the gravity globally. This would involve a direct manipulation of elements in the irrlicht engine and would allow us not to have to loop through all of the karts and update their gravity manually.

The disadvantages of using this method is that it is not certain what other game elements (currently in the game or added in the future) would be affected by the gravity reversal. It is also uncertain how kart AI would behave.

Method 2

The second method is to manually change gravity for each individual kart and maintain a cooldown of the duration globally. This would involve a boolean being set within the individual karts and thus affecting the gravity of that kart. This is safer and allows for future AI implementations to take into consideration that the kart has been reversed in gravity.

The disadvantages of using this method are that you must find a way to reference all of the karts independently without breaking the architecture. This should not be to difficult and can be done with minor references.

SAAM Analysis

Non-Functional Requirements

Performance is a Non-Functional Requirement that is not very much affected by this item as a switch in the world property would likely have minimal impact on performance. Even when looking at method two, we can avoid the overhead of looping through all the Karts by making it asynchronous and switching the gravity one by one.

Maintainability is also another non-functional requirement that is affected. By choosing method one, we run into the issue that changing gravity globally would have a significant impact on project maintainability. It introduces an area of significant possible errors that may form in the future due to other objects interacting with global gravity. With method two, however, we are more safe and the code becomes more easily maintainable as the specific karts are being changed and thus less possible errors may arise.

In both cases, we do not believe there to be much of an effect on the portability of the code by introducing this item.

Stakeholders

As a small open source project, STK does not have a very large number of stakeholders who could be affected by the change. As there is no management or investors in the project, the principal stakeholders of the project would be the developers and users.

The users will ideally appreciate an additional level of complexity, though this is dependent on appropriate testing and adjustment of the effects of the change, as a difficult to use, disorienting effect with numerous bugs will greatly detract from the player experience.

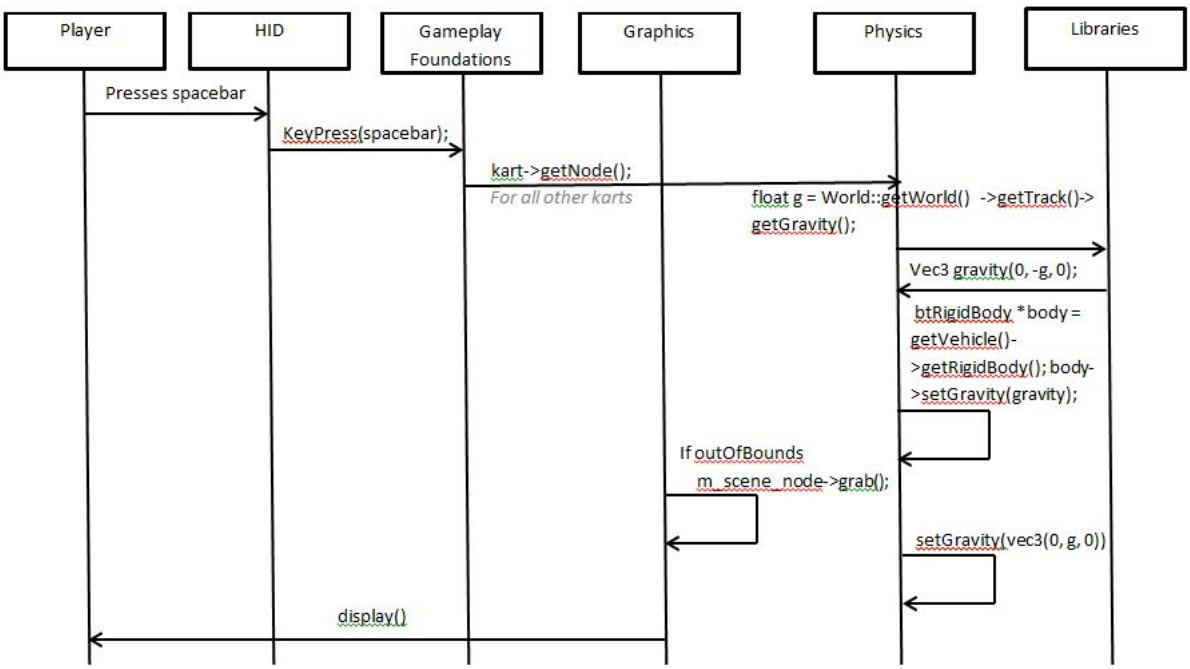
The impact of the change on the developers of STK would be variable. The change would require very little programming, as it is relatively simple and would have almost no impact on the architecture of STK. However, it is likely that the change would require substantial alterations to other aspects of the game in order to resolve bugs that would become apparent with the implementation of the item and would require a substantial amount compared with the relatively limited positive effects of the item on the player experience.

Impact on Architecture

With our proposed idea, the impact would be extremely limited. Most of the current items only have an area of effect or impact one or two karts but this new item affects all other karts at once. Adding an item in the game with a global effect may require the item subclass access to the world track properties to get the list of karts affected. If this is done, the karts will be a global object and this is dangerous. Although it would be easier to reference them globally, but it would make it very easy to change the attribute of the objects also.

Sequence Diagram

Using a Power up that Changes Gravity for Other Players



This is a sequence diagram of using a power up that will reverse the gravity for all players other than the player who used it. First, the player presses the button to use the achieved power up using space and the HID reads this input. Then we would get the nodes for all the other players on the track. After that, we get the gravity by calling getGravity() and it will return it. All this happens for three seconds after the spacebar is pressed and then setGravity() is called with the previous parameter. If the kart is out of bounds when this occurs, the referee will appear and put the kart back on track. Then setGravity() is called again with the original vector to set the gravity back to normal. Then this is all displayed for the player.

Risks and Limitations

As with all implementations, there are a several risks and limitations when attempting to implement this enhancement. First and foremost, our entire group is inexperienced with complex systems such as SuperTuxKart, and many of us have not played the game enough to know the intricacies. As a result, our limited knowledge of the details of the physics engine and libraries can make this item difficult to implement, and even more arduous when attempting to debug over several subsystems.

We will also have very limited control as to how the item will interact with existing game elements, such as not knowing which environments are suitable for this item's effect. Clipping can be a common outcome during the testing phase, and in order to fix this we will not only have to change the environment, but go through another debugging phase.

There's also a limitation of teaching the AI the proper way of interacting with the item, both when they are in possession of the item, and when they are affected. Ideally, the AI would be able to use the item's effect to influence other players, whilst knowing what to do when they see the warning. However, making changes to the AI's behaviour based on this new item can be challenging, and we understand that there will never be an AI that will use this item as well as a human player could.

Testing

Unit integration testing is a key part of any development process. It ensures that every single subsystem is complete and that new dependencies will not cause any issues. For this kind of testing, we will be testing various tasks and effects of the item.

- Firstly, we have to test the dependency on the libraries by making sure that the item is on the list of obtainable items from an item box. On top of this, we make to sure that the graphics of the item are also present on the player's screen and that it disappears when the item is used.
- The actual effect of the item, as described in the introduction. In addition to this, we will need to test players going out of bounds as a result of the effect, and players using walls and ceilings to negate the gravity reversal effect.
- As mentioned in risks and limitations, we will have to test the AI's item usage. This is is to make sure that there are no bug involving AI and this item.

System testing is the second phase of testing, where we will test the entire system. To do this, we can simply play a few games of SuperTuxKart with a mix of human players and AI. This is to track any errors that may occur on either of the types of players. Playing on different maps enables testing on different environments and boundaries such as walls and ceilings. Lastly, system testing also allows us to tweak the time of the gravity reversal effect, and change accordingly depending on feedback from testers.

Concurrencies

The addition of the new item will not change the concurrencies of the subsystems. Existing concurrencies mentioned in previous reports include HID in it's own individual thread, and graphics and sound being concurrent subsystems.

Lessons Learned/Conclusion

We noticed that there were a lot of interesting things to do. We saw that it would be helpful to look at the concrete architecture before deciding on a feature as it can show us what is feasible and possible to implement. It can also help determine what needs to be changed or if any non-functional requirements will be affected by this feature. We learned that jumping into code halfway can be tough as we need to fully understand the concrete architecture or else some dependencies could be incorrect when implementing the new feature.

With the current items in the game, our new feature separates itself from the original racing kart game. This power up can disrupt the flow of the race which can either be an advantage or a disadvantage depending on who uses it. Out of the two methods that we have presented, we went with the second method. Our SAAM analysis shows that method two would be the safer route by changing the gravity for each individual kart instead of doing it globally. With all the other kart racing games in the market, this power up can level the playing field for basic and advanced players.

Appendix

Glossary

Architecture - *Architecture is the fundamental organization of a system embodied in its components, their relationships to each other, and to the environment, and the principles guiding its design and evolution.*[5]

Component - A part of a system which performs its own set of functions.

Dependency - The need for a component to use the functions of another.

Game State - A snapshot of the game containing all of the current objects and values, if either an object or a value changes a new game state is created.

GUI - short for “Graphical User Interface” it is a visual component with which the user interacts with the software.

Multi-platform - A term describing software that can be used across many different devices

Open source - An open source project has all of its source code (and often its resources) available to the public for modification or further development by anyone.

SDK - A software development kit is a set of tools that enable creation of software for a platform or system.

Subsystem - A component created from a collection of smaller components that work as a system on their own

References

1.Gregory, J. (2009). *Game engine architecture*. Wellesley, Mass.: A K Peters.

2.Super Tux Kart Documentation: http://supertuxkart.sourceforge.net/doxygen/?title=doxygen

3.Super Tux Kart Source: https://github.com/supertuxkart/stk-code

4. SuperTuxKart development blog: http://blog.supertuxkart.net/

5. IEEE Computer Society (2000). *IEEE Recommended Practice for Architectural Description of Software-Intensive Systems: IEEE Std 1472000*. (also known as IEEE 1471)