Minigolf game

Maksym Khavil
ČVUT–FIT
khavimak@fit.cvut.cz

May 25, 2024

My goal was to create minigolf game with realistic 2D physics and different game objects affecting golfball. While creating levels for my game, I have developed easy-to-use level editing format.

1 Introduction

Me and my friends used to play mobile minigolf, while we were studying in secondary school and I had always felt lack of levels in that particular instance of golf. Despite of our many attempts to contact or influence game's representatives, we could not affect game's faith. Today I have managed to make my very own game of golf with relatively user-friendly level design instruments.

2 Game cycle

Game starts with an inviting screen and after a click user proceeds to choosing number of players. Then user can choose between playing one of two whole episodes (6 levels) or playing single level at a time. After finishing either level or episode they will be presented with scoreboard showing total score and how many strikes had each player made. On the same screen with a scoreboard players can choose to continue playing or to exit the game. If they had chosen to continue, they will be shown a level-choosing screen from the start. Scores and strikes from different levels accumulate.

3 Ball manipulation

While playing golfers need to get their ball into the hole. Active player's ball is indicated with rotating punctuated circle. They can hit it by pressing left mouse button down and dragging in the opposite direction of a hit, an arrow indicating direction and strength of a strike will spawn, the hit will be made on unpressing mouse button. After being hit the ball will move through space affected by the floors, on which it is in the given moment (more on that later), bouncing from walls and other players' balls. Last gives space to interactivity between players. When all balls stop moving, next player's turn starts.

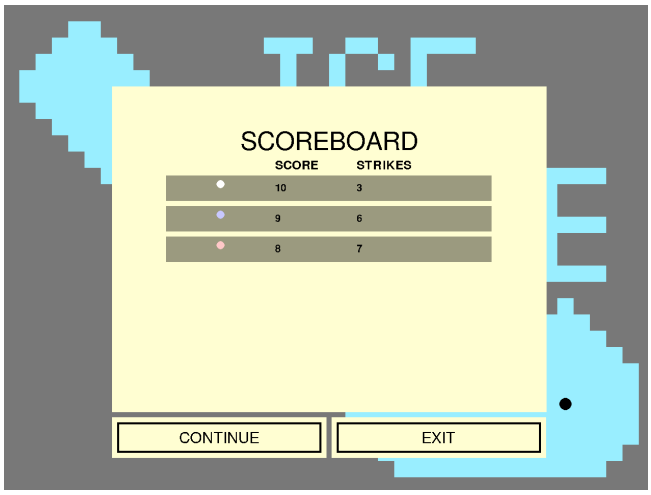

Figure 1: Scoreboard at the end of a level

When ball hits a hole, a label will occur showing how many points has player earned, and their ball is deallocated from that level.

4 Scoring

Number of points given to a player depends on number of turns, that they needed to get a ball into a hole. Players, who made it in the least turns, get the most points. After all players have reached a hole, next level or scoreboard, if it was the last level, will spawn.

5 Objects on levels

Levels have three types of flooring: grass(mediocre friction), ice (almost no friction) and sand (very strong friction). Two types of walls could be found on levels: regular walls with almost perfect bounce and silent walls that practically stop balls.

6 Methods

Game is simultaneously simulating three different parts: visual, physics and logical. They are separated due to complexity of visual part and sensitivity of physical part. At the beginning I tried implementing physics in raw python and numpy, but it

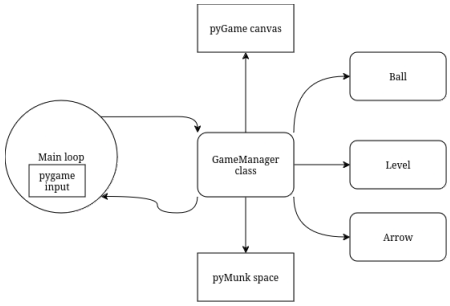

Figure 2: System diagram

failed, because balls were blinking through walls and bouncing from wrong angles. Then I have discovered pyMunk module, that implements realistic physics in two dimensions and is independent from visualising module. GameManager is both wrapper around pyGame and pyMunk and is also directing game logic. Physics simulation runs around 500 times faster than visual to minimize motion artifacts. PyMunk space provides number of variables to tweak for smother and more accurate physics, experiments showed, that while they improve rare-ocassion situations (fast-moving ball on ice bouncing from wall on a steep angle) they make altogether feeling from game much less pleasing.

7 Level creating

First several levels are implemented by hardcoding cells, which proved to be work-intense and very unintuitive, so I have created text parser that takes plain text file and generated level based on it. Source has at least 30 rows and 40 columns, where each char represents cell. Level generator parses file by such rules: 'X' is wall, 'H' is silent wall, '_' is grass cell, '+' is ice, '-' is sand. Any chars past 40th column and under 30th row are ignored.

8 Results

Game has 12 fully playable levels. Physics is smooth and very predictable, though incredibly rarely when balls are moving too fast little artifacts with bouncing, because pyMunk allows objects overlap and then ball is bouncing not from the closest wall, but from the first one processed, thus bouncing from invalid angle.

9 Conclusion

I have implemented minigolf game, I have experimented with physic and visual aspects of a game, trying to make it as good as it can possibly get. And still there is room for improvement. Due to

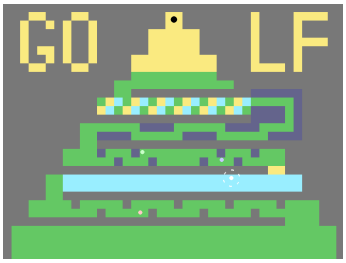

Figure 3: Level #12

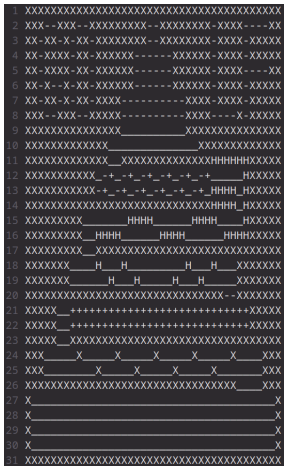

Figure 4: Source text from which level #12 is generated

sandbox nature of a game, many objects can be added such as moving walls or floors, trampoline, portals or enemies. Also LAN or net multilplayer will be an amazing addon for a game like that.

References

[1] Coding with russ. `https://www.youtube.com/@CodingWithRuss`.

[2] pygame: python gamedeveloping kit. `https://www.pygame.org/docs/`.

[3] pymunk: python physics engine. `http://www.pymunk.org/en/latest/index.html`.

[4] pytest: python testing utility.

[5] Software diagram drawer. `https://app.diagrams.net/`.