River and Frog Arcade Game using pthread

Xin Huang
Dept. of CST, THU
ID: 2011011253

August 30, 2013

Abstract

A River and Frog Arcade Game implemented with gtkmm and pthread.
This is *homework 2* for course *Parallel Programming*

Keyword River and Frog Arcade Game, pthread

Contents

1 Instruction

1.1 Prerequisite

This program uses *gtkmm-3.0*(along with glib and gdkmm) and *pthread* as its back end, so you **MUST** have them installed.

1.2 Compilation

Invoke *make* to compile the source code. Also you can set environment variable *DEFINES="-DUSE_PTHREAD"* to enable pthread functionality. Executable is sorted in *bin/river-and-frog*

1.3 Execution

Invoke

make run

to run with the default setting.

For advanced options, see section **??** You can stretch the window for bigger size.

2 Introduction

You are to control a frog jump over the river by stepping on wood floating in the river. The frog dies if it jumps into the river(maybe crocodiles waiting with hunger).

3 Control Setting

- In single mode, Player 1 control the frog using arrow keys.
- In two player mode, Player 1 use arrow keys, and player two use *w, s, a, d* keys.

4 Programming Design

- The program is written with Gtkmm and pthread. This program has Game, Layer, Wood and Frog. Game deals with the start and update. Layer calculates the speed and position of each wood on the layer. Wood loads the image and grasp the position of the frog(is the frog on the wood). Frog works for the movement of the frog.
- I use Glib::timeout to refresh the Game and the canvas. Game is the conductor. Every time the Game::update executes the speed and position of woods and frogs will be computed and update the canvas.
- Multithread programming is used in Layer. The number of threads is same to the number of layers. Each thread computes the collision on its own. When all collision computation finished(use pthread join to block), the update function will move the wood and the frog. And then the update loop is finished.

  - Game controls Layer, Layer controls Wood, Wood controls Frog.
  - Game::start() first invoke Game::init() first to initialize the canvas, woods and the frogs. Then invoke sigc::mem$_{f}unandGlib :: signal_{t}imeouttorefreshtheGame ::$
$update().TheGame :: update()useatimertocounttheremaintimetolettheLayercalculatetheeverylo$
$update()isrunineachthread.Everythreadwillgetalayerandcalculatethewoodscollisionsonitslayer.$

- Each time the Layer would move only one wood that will immediately collides with another. The collision form is elastic collision. When the remain time is run out, the thread will wait for other threads to stop.
- When Layer::update() finishes the Game::update() will move the frog. The frog can move to up, down, left, right. The message is deal with Gdk Keyval.
- The frog will go up to the next layer if user presses left key. The Frog object will be removed from the current Wood and jump to (add to) the upper Wood. But if there is no woods, the frog will be removed and added to the basement.

5 Features

- River consists of multiple layers, wood slabs can only move in one layer.
- Wood slabs have masses proportional to its width.
- By default setting, when two woods hit, the resulting speed of the wood slabs obeys the conservation of momentum and energy.

  Watch out if a large piece of wood slab's about to hit you!
- Wood slabs are in different size and moving in different speed. The wood slabs in last layer moves really quick, so be careful when you move
- Frog can jump between wood slabs in two different layers, but can not change to wood slab in the same layer.
- The upper the wood slab, the faster it will be.
- There is a slight acceleration in frogs moving speed, and frog will stop as soon as you release the control keys.

  Get across the river as fast as you can!

6 Specification of Pthread in Program

Pthread is used for calculate collisions between wood slabs. During each frame, each layer is assigned a thread to calculate collisions simultaneous.

7 Experience

Writing a GUI program with Gtkmm is not so convenient. I wrote the program in the first time, in which I need to look up the documentations and ask classmates for help. When I have debug it for two day and nothing changed, I rewrite the program. Though the experience is tough and not very pleasant, I now know how to write a Gtk program with pthread. And now I realize it's also important to schedule the time. In the first place I want to write a n-body simulation program and switch to this task, but it's too hard and I waste two days. Fortunately I change my idea and I can finally finish this task. Preparing and scheduling are now have to be taken in my consideration.

8 Screenshots

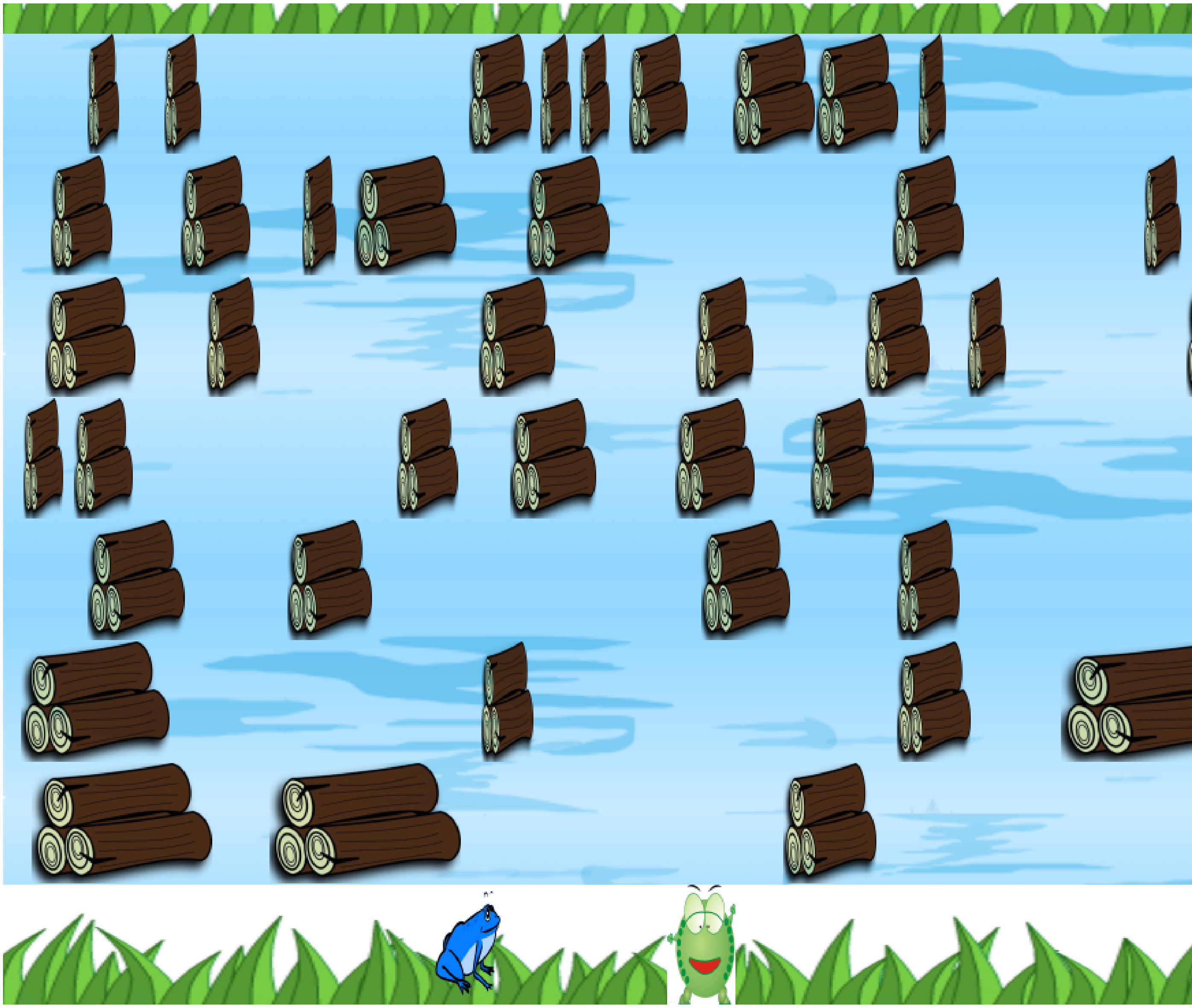

Figure 1: Game start

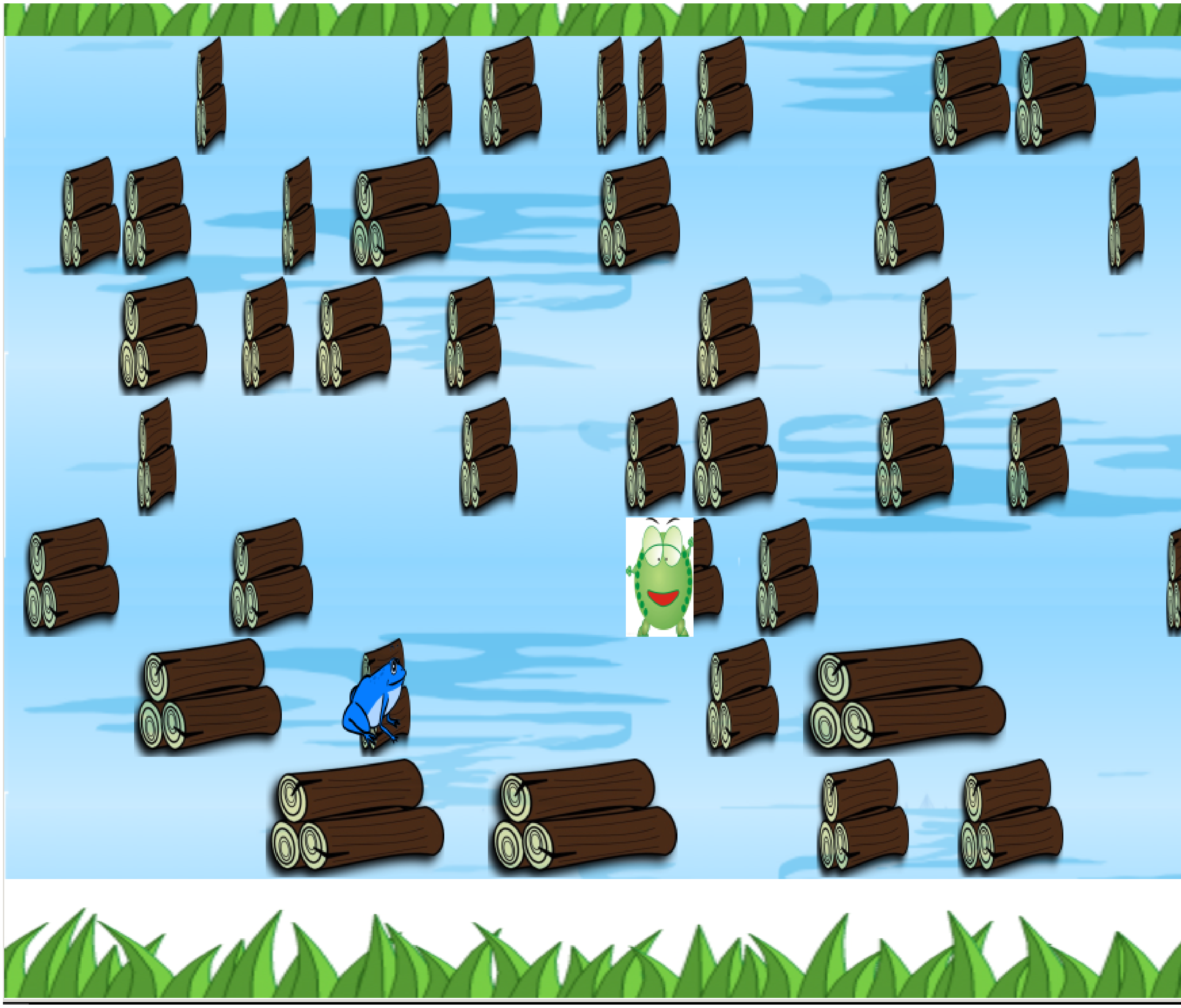

Figure 2: During game

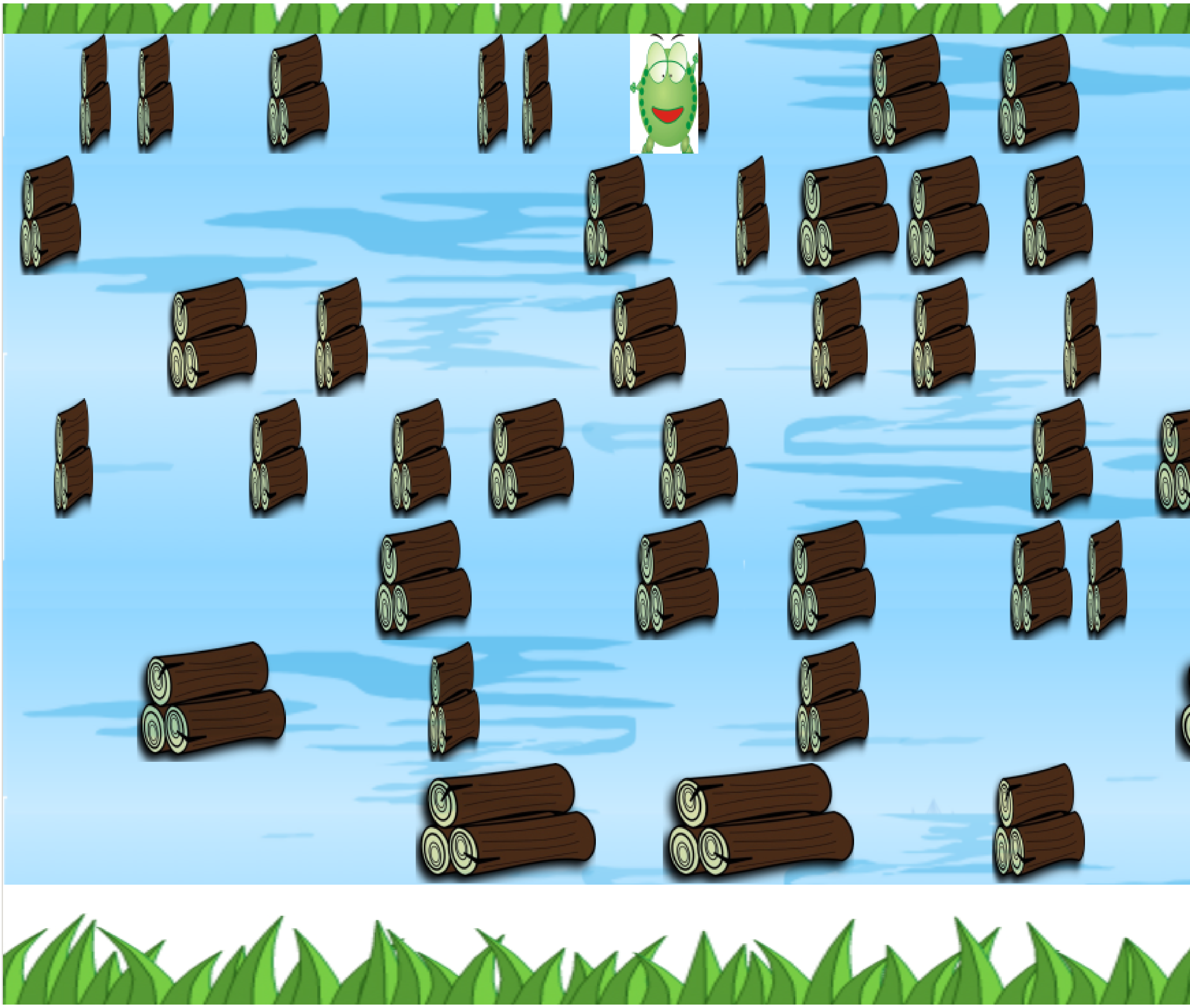

Figure 3: Game with two players