ENG2002 Computer Programming

Mini-project 6 Train Routing

Final Report

Group No: 29

Name: PENG Tao, ZHAO Tianyi

Student ID: 13104616D, 13103359D

Catalogue

Abstract

This project concerns a Train Routing Game design and implementation by using C++ computer language in Visual Studio 2010. There are two major design environments utilized in VS2010 which are Win32 Console Application and Windows form Application. In Win32 Console Application, static libraries of game classes with varieties of member functions are written and tested to achieve nearly all of the demands in the game. While in Windows form Application, Graphic User Interface is developed based on these libraries to display the game and interact with players.

An integrated game is completed after the whole processes of design and implementation. The game is designed in principle of user-friendly interface, interesting game logic and artistic appearance. The GUI is designed in a concise and geometrical style.

Introduction

Game description:

Train Routing is a game to control the routing of a railway network for the trains to travel from the starting point to the destination. Player can routing by switching the joints to select the connections between joints. The starting points are located at the bottom-right station with two tracks while the destination will be selected at random from other stations.

Project objectives:

To design an interesting and reasonable game logic.

To implement a user-friendly and nice-looking game interface.

Project requirements:

a. 10 joints are required to be displayed in the game, which can be controlled by the player.

b. 3 destination stations of different colors are randomly selected as the destination of a train, which is in the same color of its destination.

c. One roadblock is built on a track, which can reverse the move direction of train when the train runs into the roadblock.

d. The score can be calculated for the player when a train gets its required destination within given time.

e. Before the player start to play, he needs to login with a username and correct password. After playing the game, the player's highest score can be saved in a file.

f. Two additional features:

   i. Ghost train: a new type of train with no destination at the beginning. After several seconds, it will be assigned with a random destination. Before that, the arrival at any destination will cause a wrong destination error.

   ii. Bonus item: a special item located on a random track, which worth 100 points. The players can collect it by routing any train passing through it. Whenever it is collected, a new bonus item with random location will be generated.

Methodology:

Work distribution:

Stage \ Student	PENG Tao	ZHAO Tianyi

Plan and prepare	Decide what classes to build;
	Distribute jobs;
	Design the route;	

	Construct whole program framework;	Search for information and materials;

Classes build & functions implement		Communication with each other;	

		Build Joint, Track, Transform class;
		Integrate GameObject classes;
		Write testing program for GameObject class;
		Debug and revise the program;
			Build Train, Cab, RoadBlock class;
			Log class;
			Write testing program for Login class;

GUI design	Framework	Art direction;	Design of 3 forms;
			Components in 3 forms;
			Relationship of 3 forms;

	Game form (Form1)	Train animation;
		Glowing effect;
		Program for random;
		Background paint;
			Draw components;
			Pause, restart and quit function;

	Mainmenu form	Background paint;	Animation of buttons;
			Write texts;

	Login form		Login system;
New features		Bonus item	Ghost train

Improve & debug		Test game;
		Collecting feedback from others;
		Modify game parameters;	

Schedule:

Feb. 20	Planning the project;
Feb. 25	Discussing the classes distribution; Starting to write game related classes;
Mar. 2	Completing the framework of classes;
Mar. 14	Finishing the GameObject classes;

	Writing test program;
	Starting GUI designing;

Mar. 18	Finishing the testing program; Writing interim report;
Mar. 20	Finishing the framework of GUI design; Finishing the interim report;
Mar. 27	Finishing train animation; (First version completed) Design of main menu and login form; Writing Log class;

Apr. 4	Finishing Log class;
	Finishing main menu and login form; (Second version completed)
	Designing the route;

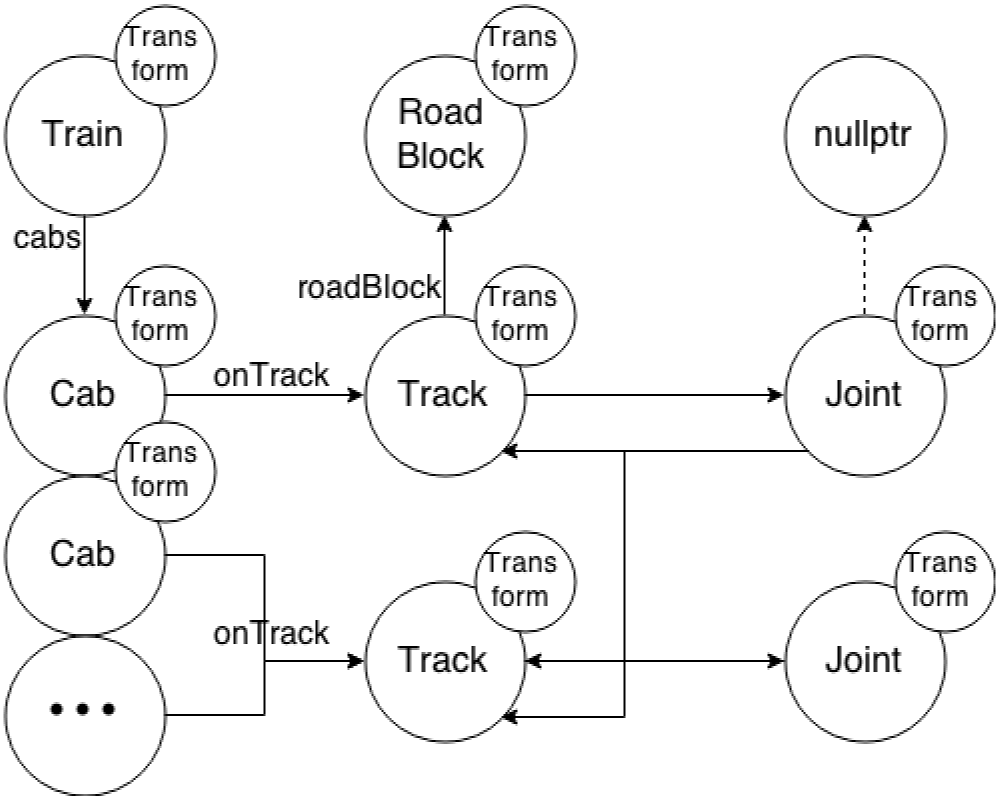
Apr. 9	Optimizing game logic;
	Debugging for the 3 forms;
	Starting artificial design;

Apr. 13	Debugging;
	Finishing the design of route;
	Implementing new route;
	Adding roadblock and traffic lights; (Third version completed)

Apr. 14 –
Apr.17
	Debugging;
	Optimizing game logic;
	Beautify the interface;
	Adding two new features;
	Collecting feedbacks from other students;
	Revising again and again; (Forth, fifth and Beta version completed)
	Writing final report;

Specification:

Class relationships



class Transform

/*

A Transform class is the basic element for every game object. This class will store the position and rotation of every game object. The position properties can be used in GUI programming: x and y can be directly used as coordinate in GDI drawing. Also, some convenient functions are provided to simplify the calculation code.

*/

Methods:

public:

/*Constructors and Destructors*/

```
//Default values of constructor were set to origin
 Transform(float X = 0.0f, float Y = 0.0f, float Rotation = 0.0f);
 ~Transform();


 /*
  Following three functions will provide convenience for calculation code in game programming.
 */
 float DistanceTo(Transform target);
 float DistanceTo(float X, float Y);
 float AngleBetween(Transform target);


 /*
  To modify the transform easier, Transform class provides some member functions similar to human thinking way.
 */
 void Move(Transform DeltaVector);
 void Move(float Dx, float Dy);
 void MoveTo(Transform toTarget);
 void MoveTo(float X, float Y);
 void Rotate(Transform target);
 void Rotate(float Dr);
 void Scale(float Factor);
 Transform Normalize();
 Transform Intersection(Transform t1, Transform t2, Transform t3, Transform t4);
```

Fields:

```
private:

float x;

float y;

float rotation;
```

---

class Joint

```
/*

 Joint class is the object for represent track joint. Joint will store the pointers to

 linked joints and tracks in the form of their pointer. Joint also have switch property,

 by using the Next() function, it will return the pointer to next track under current switch
     condition.

*/
```

Methods:

public:

```
/*

 Get next track under current switch condition

*/

Track* Next(Track* FromTrack);

////***Constructor***//

/*
```

```
 Different constructors are design for various condition when creating joint object.
*/

Joint();
/*End Station Constructor*/
Joint(Joint* NodeIn, Transform Transform);
/*Begin Station Construcotr*/
Joint(Joint* NodeOut0, Joint* NodeOut1, Transform Transform);
/*Joint Constructor*/
Joint(Joint* NodeIn, Joint* NodeOut0, Joint* NodeOut1, Transform Transform);

~Joint();

void setjointID(int JID);
int getjointID();
//Properties accessors
void setNode0(Joint* Node);
void setNode1(Joint* Node);
void setNode2(Joint* Node);
Joint* getNode0();
Joint* getNode1();
Joint* getNode2();
```

```
 void setTrack0(Track* Track);
 void setTrack1(Track* Track);
 void setTrack2(Track* Track);
 Track* getTrack0();
 Track* getTrack1();
 Track* getTrack2();

 //Switch
//Lock the joint by the cab's ID so that when a train passing the joint, it cannot be change switch.
 void changeSwitch(bool TS);
 void inverseSwitch();
 bool getTS();
 void setTS(bool TS);
 void lockTS(int CabKey, int CabKeyN);
 void unlockTS(int CabKey);
 Transform transform;
private:
 int jID;
 //Pointers to store the linked joints and tracks
 Joint* pnode0;
 Joint* pnode1;
 Joint* pnode2;
 Track* ptrack0;
```

Track* ptrack1;

Track* ptrack2;

/*

There will be a possibility that if the player change the state of joint when the train only partially pass the joint. The train will be seperated into two. We add an tsLock to prevent such bug.

*/

int tsLockA;

int tsLockB;

bool ts;

//ts trackSwitch stores the switch state

/*

Switch: false: Node0<->Node1

true: Node0<->Node2

*/

class Track

/*

Track class is similar to joint class. It will store pointers to the

linked joints. Next() function can return the the pointer to the next

joint. Track class will store whether the track has roadblock or not.

*/

Methods:

public:

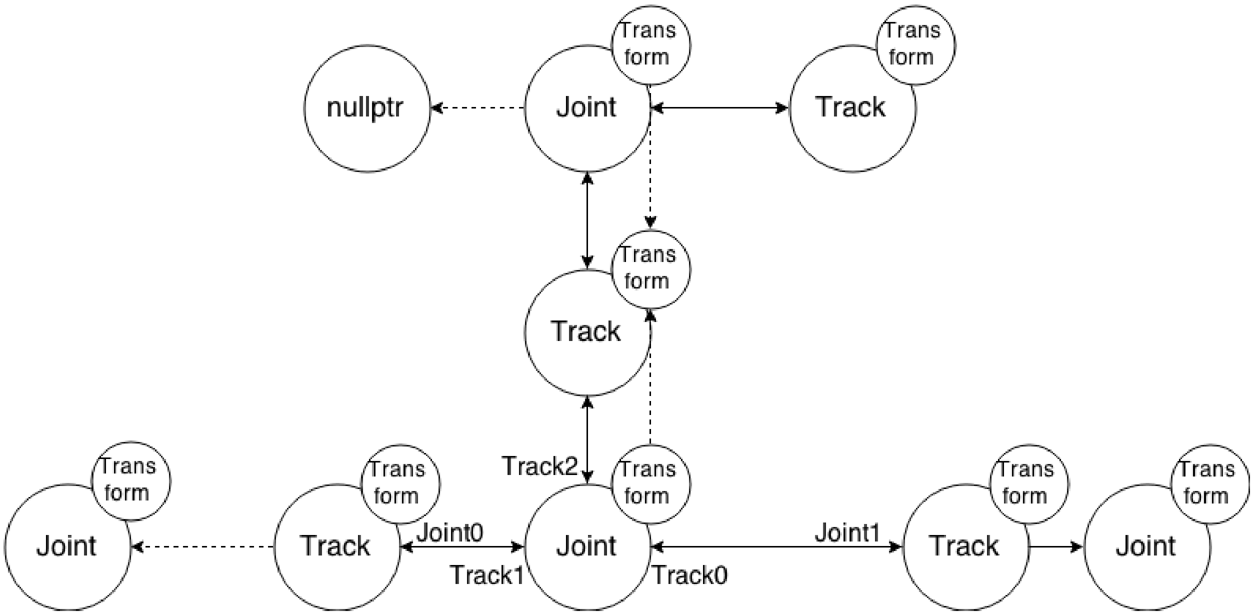
```
Joint* Next(Joint* From);

/*

 setupTransform will calculate the length and rotation of the track and

 store it in member fields.

*/

void setupTransform();

//Constructors

Track();

Track(Joint* node0, Joint* node1);

//Destructors

~Track();


//Properties accessors

void settrackID(int TID);

int gettrackID();

//Joint { set; get; }

void setJoint0(Joint* Node);

void setJoint1(Joint* Node);

Joint* getJoint0();

Joint* getJoint1();

//Length { private set; get;}

float getLength();

//hasRoadblock { set; get; }
```

```
 void sethasRb(bool HasRB);

 bool gethasRB();

 //roadBlock { set; get; }

 void setroadBlock(RoadBlock* RoadBlock);

 RoadBlock* getroadBlock();



 Transform transform;
```

fields:

```
private:

 int tID;

 float length;

 Joint* joint0;

 Joint* joint1;

 bool hasRoadBlock;

 RoadBlock* roadBlock;

/*

    Use hasRoadBlock to reduce the calculation, otherwise, whenever a train travel through the track,
    it has to calculate a useless distance for several time

*/
```

In general, Joint and Track are connected like this diagram:



class Train

/*

Train class is the object generally control the train, it contains several cabs object. Children object cabs are stored as pointer of array. Train object will not be directly display in game scene, but the cabs will be displayed. Train can move the cabs by passing the speed and delta-time between frames as arguments to the Move() function in cab object.

*/

Methods:

```
public:

int Move(float DeltaTime);

 void Reverse();
 Train();

 /*Construt a Train object with 'cabsNumber' cabs*/

 Train(int cabsNumber, Transform transform);


 ~Train();



 void settrainID(int TID);

 int gettrainID();


 void setdestination(int Destination);

 int getdestination();


 Cab * getcabs();


 int getcabsNumber();



 void setSpeed(float Speed);

 float getSpeed();
```

fields:

```
private:

 int tID;
 int destination;

 Cab * cabs;

 Transform transform;

 int cabsNumber;//To store the total number of cabs, used in for loop

 float speed;

 bool inversed;
```

---

class Cab

/*

Cab class is the basic element of game object, it can move and rotate, they will be driven by train object. During most of time, this class will not be directly accessed.

*/

Methods:

public:

```
int Move(float Speed, float DeltaTime);

//Constructors

Cab();

//Destructors

~Cab();



void setcabID(int CID);
```

```
 int getcabID();


 void settrainID(int TID);

 int gettrainID();


 void setonTrack(Track* OnTrack);

 Track* getonTrack();


 void setlastJoint(Joint* Joint);

 Joint* getlastJoint();


 void setcabLength(float CabLength);

 float getcabLength();


 void setcabNumInTrain(int CabNumer);

 int getcabNumInTrain();
```

fields:

```
public:

 Transform transform;


private:

 int cID;

 int tID;
 int cabNumInTrain;

 Track* onTrack;

 /*

  LastJoint pointer will store the passed joint and be used to retrive next track

  form next joint's member function Next()

 */

 Joint* lastJoint;

 /*

  cabLength is the interval between cabs, it will be used in train initialization.

 */

 float cabLength;
```

---

class RoadBlock

```
/*

 RoadBlock specify the roadblock element in game. It will store the radius and

 location in the track. It use onTrack pointer to store which track this roadblock

 is on.

*/

public:


/*

 SetupTrasform() function will calculate the specific location (x y coordinate in

 canvas).
```

```
*/
void SetupTransform();
//Constructors
RoadBlock();
/*
 Location is the percentage position on the track, therefore, GUI programmer can
 locate it easier. Radius will help the collision calculation to be more accurate
 and natural.
*/
RoadBlock(Track* onTrack, float Location, float Radius);
//Destructors
~RoadBlock();



void setroadBlockID(int RBID);
int getroadBlockID();

void setonTrack(Track* OnTrack);
Track* getonTrack();

void setlocation(float Location);
float getlocation();
```

```
 void setradius(float Radius);

 float getradius();


 void setblockState(int BlockState);

 int getblockState();


 Transform transform;

private:

 int rbID;

 int tID;

 Track* onTrack;

 float location;

 float radius;

 int blockState;

/*

        The variable blockState will indicate the property of this road block, 0 for red traffic light, 1 for
       greenlight or other passing block, -1 for road block that will turn train around.

*/
```

```
    class Log
    /*

              This class is used to establish a file for user accounts, user identification, scores storing.

              A file named "UserInfo.txt" stores username, password and highest score of a user.

    If the input username is not find in the existing file, a new account is established.

    If the input username exists and the password is correct,the highest score will be updated.
```

Each account contains 4 lines:

a line with "p" standing for a existing account, a username line, a password line and a score line

like this:

```
* * * * * * * * * * * * * * * * * * * * * * *
*        p                              *
*        player1                        *
*        player1password                *
*        00200                          *
*        p                              *
*        player2                        *
*        player2password                *
*        00000                          *
* * * * * * * * * * * * * * * * * * * * * * *
```

*/

Methods:

```
public:
    Log();
    ~Log();
```

/*

When user finishes the input, exam the user information:

return -1 if not existing

return 0 if existing but password is wrong

```
    return 1 if existing and password is correct, then put the cursor position in frount of the score.
    */
    int existsUser();

    //When the input username not exists, establish a new account to the file.
    void newUser();

    //When the user finish a game, write the higher score.
    void wirteHiScore();

    //Return the score stored in the file.
    int readHiScore();

//Properties accessor
    void setPos(long);
    long getPos();
    void setFilename(char*);
    char* getFilename();
    void setUsername(char*);
    char* getUsername();
    void setPassword(char*);
    char* getPassword();
    void setScore(int);//This function also write scores into "score5" as char[].
    int getScore();
```

Fields:

private:

long pos;//Stores the cursor position at the beginning of the score.

char *filename;//Stores the current filename.//*****Usually not changed*****//

char *username;//Stores the input username.

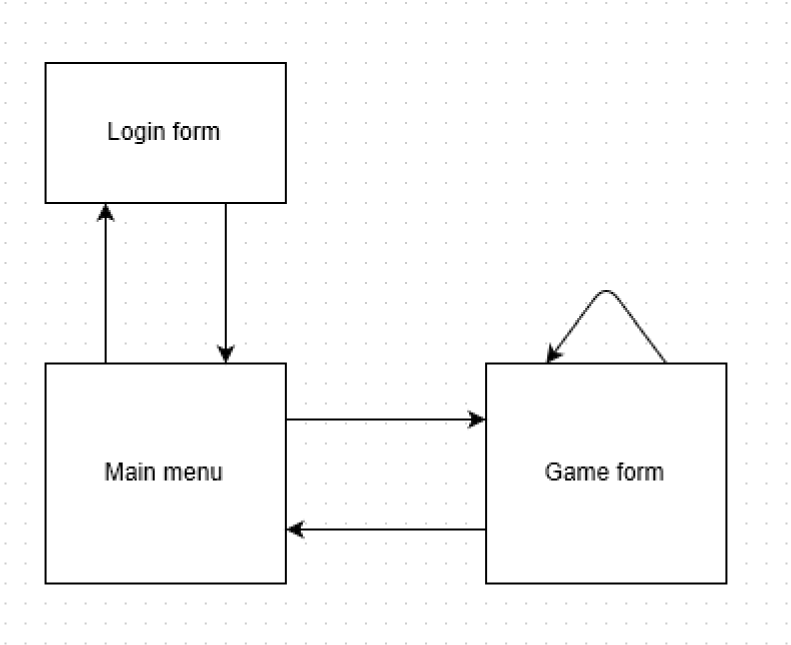
char *password;//Stores the input password.

int score;//Stores the game score as int.

char score5[6];//Stores the game score as char[] of 5 charactors.

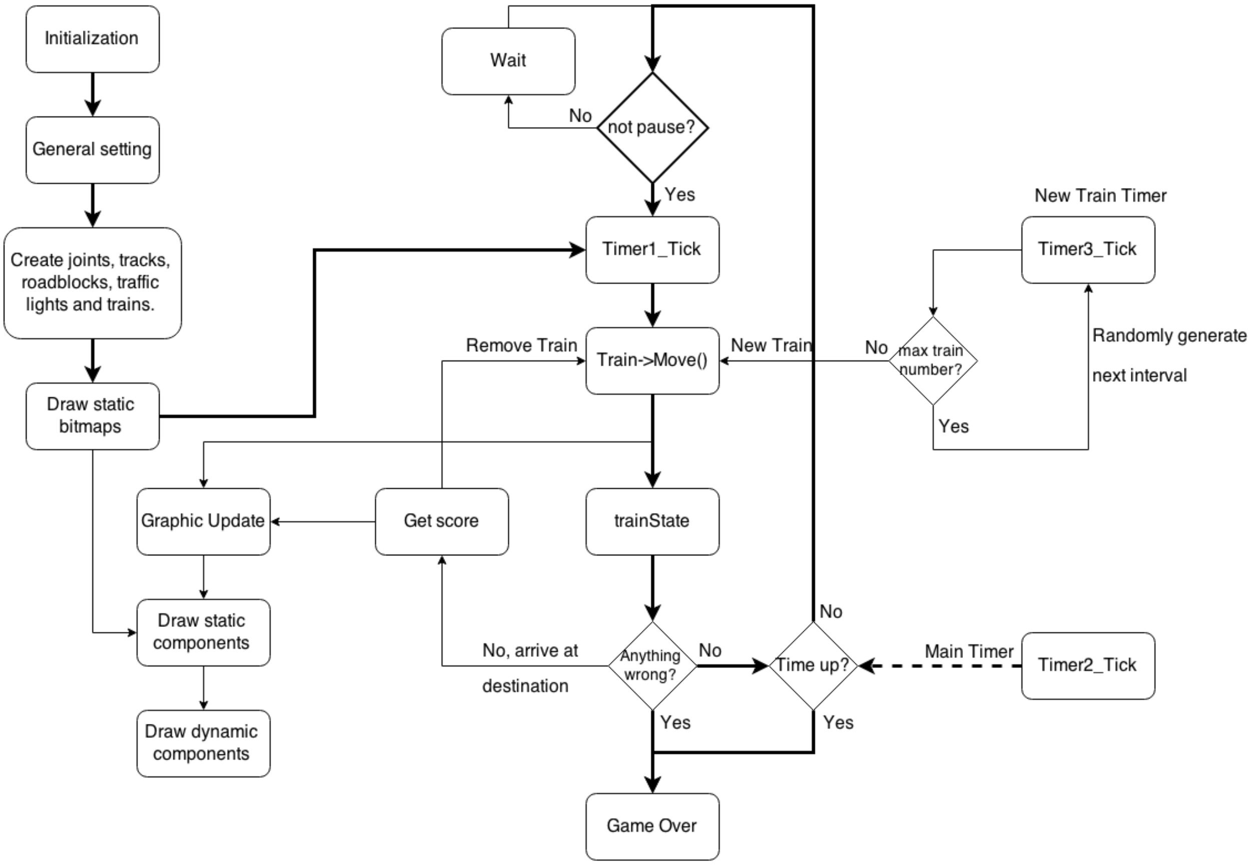
The flow of execution

Form relationship:

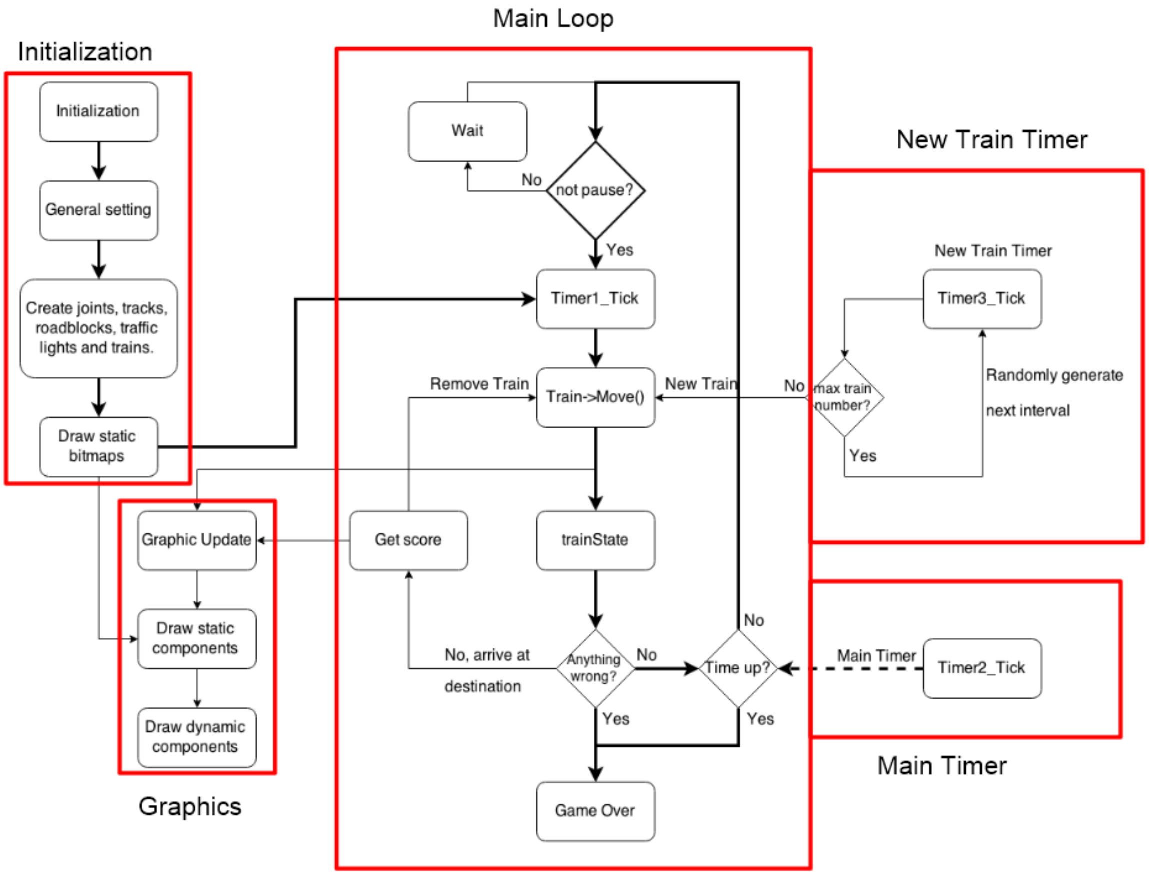


The Main menu will be the entrance of our program, and then it will call “Login” form first to ask player to enter their username and password. The main menu can start the game by creating a modeless game form “Form1” to start the core game part. The game form can restart itself or return to main menu.

Game flow:



And their functional categorization



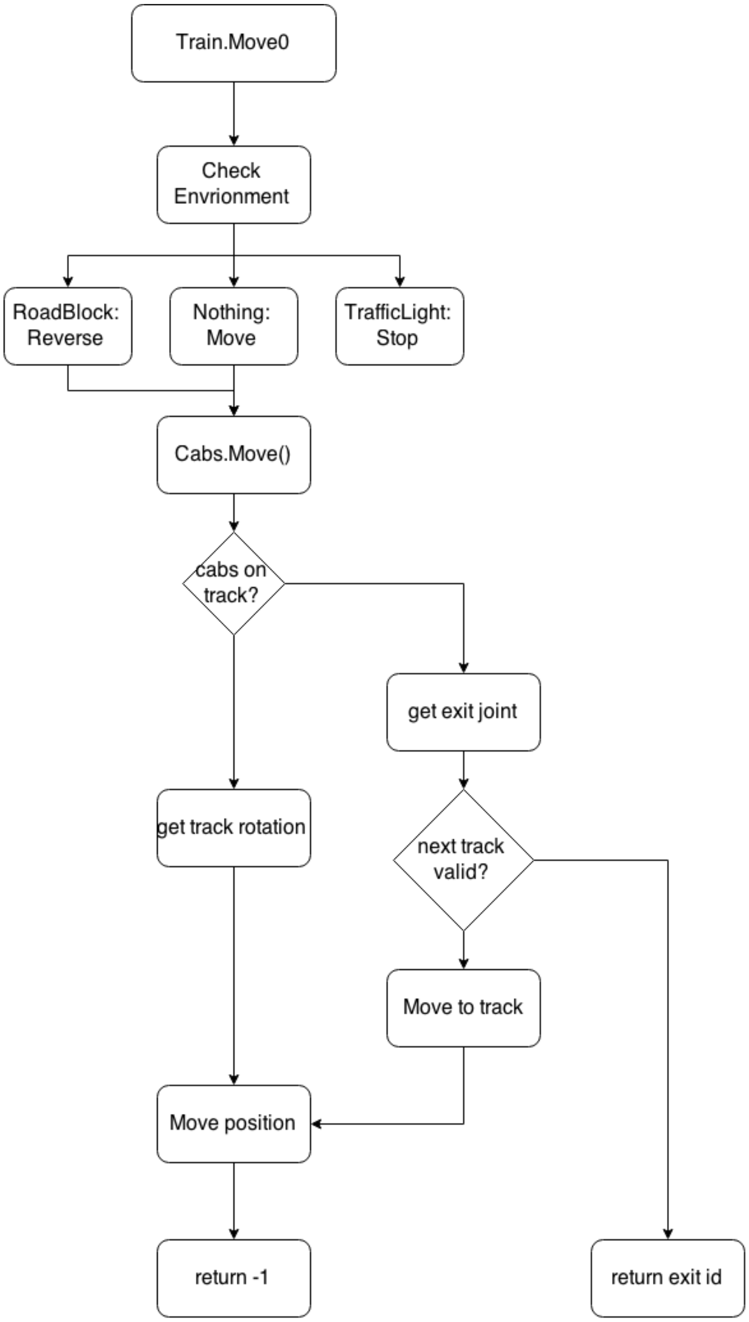
The thicker arrow lines are main flow, the thinner arrow lines are sub functions.

Generally speaking, it will do

1. Initialization
2. Create static bitmaps for graphic update
3. Drive train
4. Do graphical update according to current status
5. Detect any event(Arrival at destination, Crash, Time up) and deal with events
6. Check the game over condition
7. If the game is not over, go back to 3. And repeat.

Core train driving system:

"Train.Move()" is the core function of the game.



Train.Move() will do

1. Check the environment
2. Deal with any detected condition(encountered roadblock or traffic light), that is modify the train's state
3. According to the train's' state, call the Move() function of each Cab object
4. Cab.Move() will do

    a. Get the rotation of track which it running on.
       //cabs->getonTrack()->transform.getR()
    b. Move itself by modifying the Transform object using the methods provided in Transform class//
       //Such as cabs->transform.MoveTo(), Rotate())
    c. Detect if it is still on the track
    d. If run out of track, determine which exit joint of track is.
        i. Get next track pointer from Joint object function "Next(Track*)"
        ii. Move on to that track, and modify onTrack pointer.
       Or:
        iii. The next track is empty(disconnected joint switch or end station)
        iv. Return the id of exit joint to caller
    e. else return -1(Normal state) to caller

5. If no Cab return error, return -1(Normal state) to caller
6. Else return the id of exit joint of any Cab to caller to check condition.

Four Timers

Timer1: Interval 10ms, drive the train.

Timer2: Interval 1s, count the game time

Timer3: Interval random, create new train.

Timer4: Interval random, assign destination to ghost train, it will only tick once.

Create new train

Timer3_tick will call RespawnTrain() function.

RespawnTrain() requires a parameter of train object pointer. It will set up all the properties of train. And then, it will return this pointer and store it in the "train" pointer, which is a pointer to an array of pointer to train object. CurrentTrainCount variable will tell the first available empty pointer slot. Hence, as long as there free memory, we can have unlimited number of trains on the map.

Timer3_tick will also randomly generates the next interval, which results in random time interval of creating trains.

Delete train

DeleteTrain() function will find the train which the parameter pointer points to, and then delete the train and assign the nullptr to it to mark it as empty. This nullptr will be used when other code want to access the train pointer. Finally, it will move the remaining train pointers up to fill the empty slot, which is better for the "For loop" operation in other place

First new feature – Ghost train

Ghost train is generated as a no destination train. We modified the RespawnTrain() function. Considering the difficulty of green train and ghost train, we add sub random decision maker if the train is set to green train. So, the totally possibility of ghost train is 1/9. The ghost train's destination is set to an unused joint. Then the timer4 is enabled with an interval of several seconds. The timer4_tick will randomly set the real destination of the ghost train.

The color is updated every paint event, it will automatically draw the color of every train according to its destination.

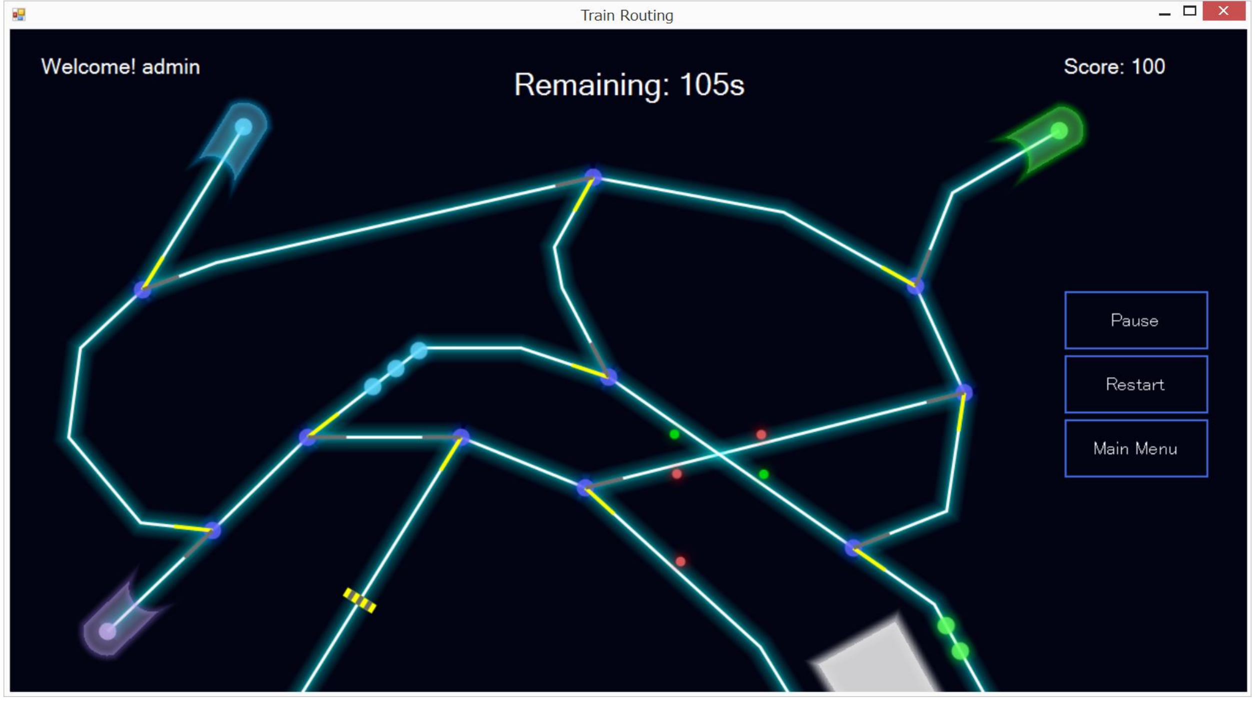
Second new feature – Bonus item

Bonus item is generated as RoadBlock class object. Its blockState will be set to 1, which means it will not stop trains. The detection will be done in CollisionDetect(). The CollisionDetect() function will calculate the distance between every two-combination of all the cabs and bonus item. If the train collide with bonus item, 100 points will be added and the bonus item will be created again at random location. It is done simply by "delete" the bonus item pointer, and then "new" a new one.

User input:

If user clicks the game interface, it will trigger Form1_MouseDown event, then we will check the controllable object one by one, calculating the distance between objects and click position, to determine which object is clicked. Also the distance must smaller than a certain value， for instance, 30 pixels in our program, to prevent ghost click;

Graphic Flow



Sample screenshot

The program will create static bitmap for storing the static components such as track and station. Then the paint function will draw the static image first and then get data from game objects to draw dynamic object such as trains and joint.

Other UI use the feature provided by winform. Just adjust the properties of components to realize our design.

Game Logic Control

The game will over if any one of follow condition is true:

1. Run out of game time(120s).

2. Every timer1_tick, CollisionDetect will run once to tell whether there is a train collision.
3. Every Move() function will detect whether the train is crashed on the disconnected joint switch.

Problems:

1. The program will crash if the train arrives at destination or crashed on a disconnected track

At very early development stage, when the train arrives at destination or crashed, it will be moved on a nullptr. The program will crash if access a nullptr. So, we added some code to delete the train if it was moved on a nullptr. In another aspect, whenever the code will access a pointer, it will detect if the pointer is a nullptr or not before use the contents of it.

2. Cannot draw multiple cabs for one train

The cabs are stored in a pointer called “cabs”. We change the pointer from a pointer to a Cab object to a pointer to an array of Cab object. This can also reduce the modification work of code. We can still use the cabs-> to access the first Cab object.

3. Some bug when train running inversely after bumped into a roadblock

At first, the Train.Move() function will do some environment detect and condition check, it will calculate from the first cab to the last cab. The order should be inversed if the train run reversely. We added a bool variable called”inversed” to deal these two condition.

4. Multiple trains cannot run simultaneously

Similar to the multiple cabs problem, our trains are stored in a pointer called “train”, we change it from a pointer to Train object to a pointer to an array of pointers to Train object.

We use array of pointer is because that the train will be dynamically deleted, so there will be some empty pointer if we did not use array of pointers. Both methods can rearrange the pointers to place then one following one. However, when other codes go through each train for some data, it needs go through all trains and check whether it is valid or not, but it cannot use for loop because it is discontinuous. Array of pointers can use index to access given number of pointers, while pointers cannot.

5. Cannot automatically create new train

We added another timer. After we changed the structure of trains' pointer, we can dynamically create a new train at the first empty slot of "train" pointer. The new timer will tick every few seconds, and it will randomly set the next ticking interval so that the generating interval will be randomized.

6. The framerate of our game is very low

As for we added glowing effect to the game, there are a lot of codes to draw the simple shape repeatedly. We use a Bitmap to store the basic elements of components. The program will only create it once and use it for following drawing.

For instance, a glowing circle needs 14 times circle drawing with different radius and opacity every 10ms. If we draw it on a bitmap at first, and then we only need to draw a bitmap once every 10ms. It reduces the calculation significantly.

7. The trains will not collide with each other

We added a collision detector function and execute it every time after the train is moved.

8. If the user change the switch of joint when the train only partially pass the joint, the train will be separated into two

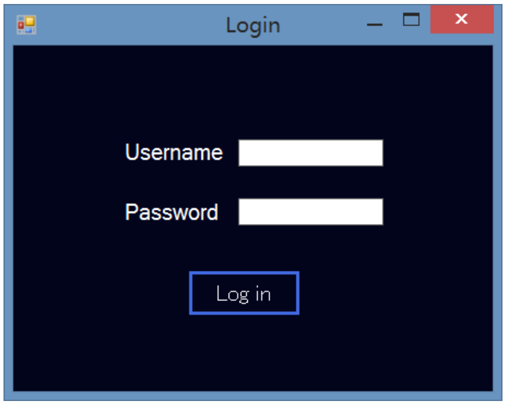
We add lock and unlock method to joint. If the joint is locked, its switch cannot be modified. When a cab passes an unlock joint, it will calculate the id of another end cab as the key to lock the joint, so only the last cab of the train can unlock the joint. The id of each cabs is unique starting from 0.

9. A traffic light is needed to control the train at a certain position

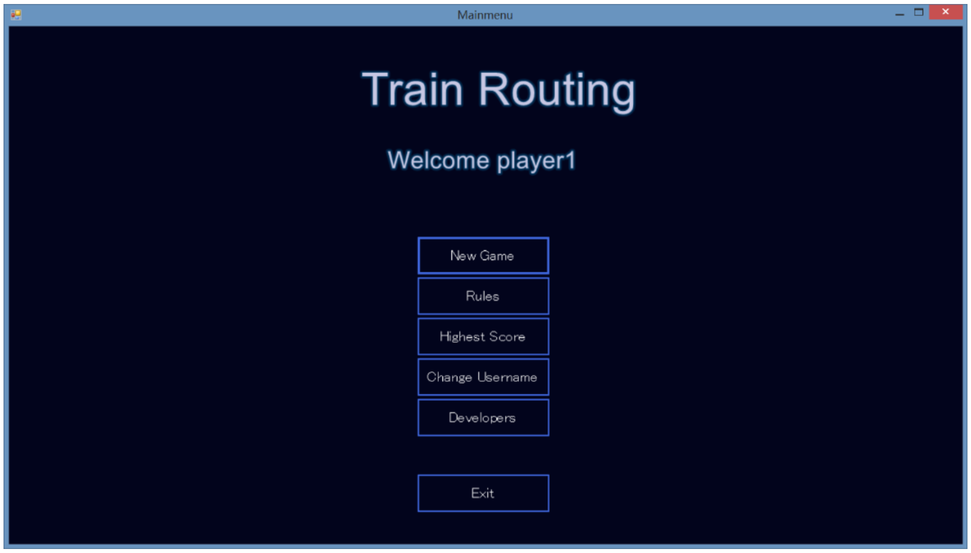
Considering the similarity between traffic light and roadblock and the process of train.Move(), we modified the RoadBlock class and add another field called “blockState”, it has three possible value: -1 for roadblock, 0 for red light, 1 for green light. The train.Move() will know from the cab whether there is a roadblock object on that train or not, and then it will calculate the distance between them. If the distance is smaller than a certain value, it will do some modify on train state. So, the traffic light is achieved at minimum code modification.

Results:

The Login form:

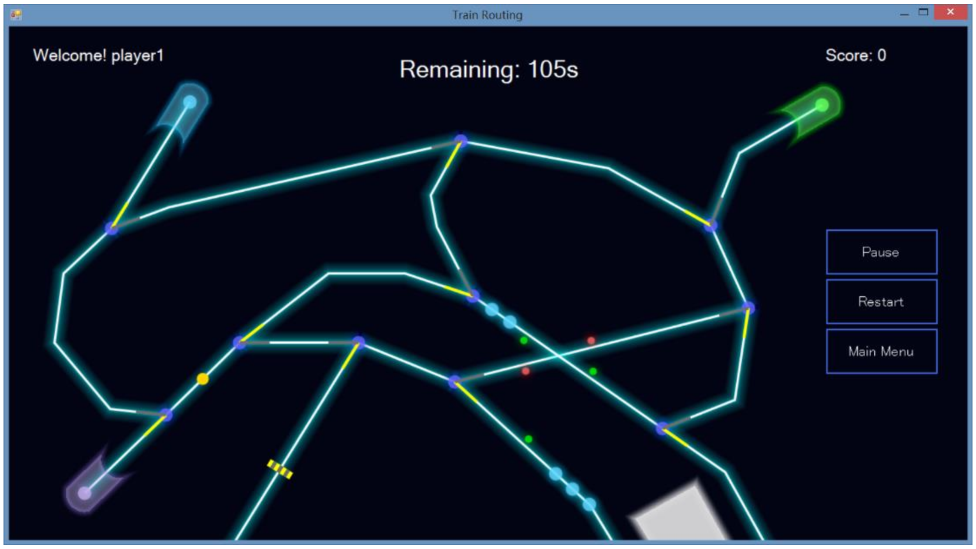


The main menu:



Welcome player1

Game running:



Welcome! player1

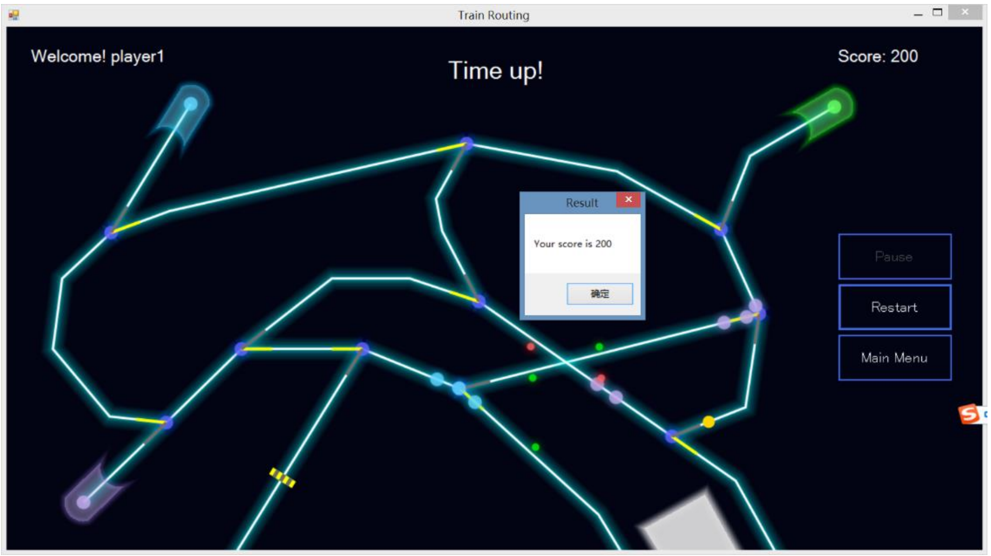
Remaining: 105s

Score: 0

Game Over:

The game will come to an end when one of the 3 situation happens:

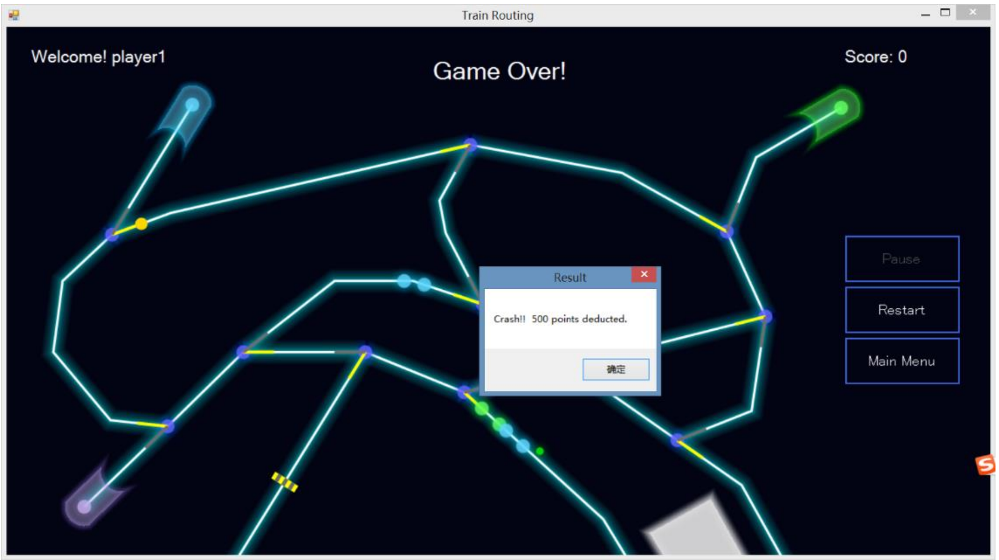
1. Time is up.



Result

Your score is 200

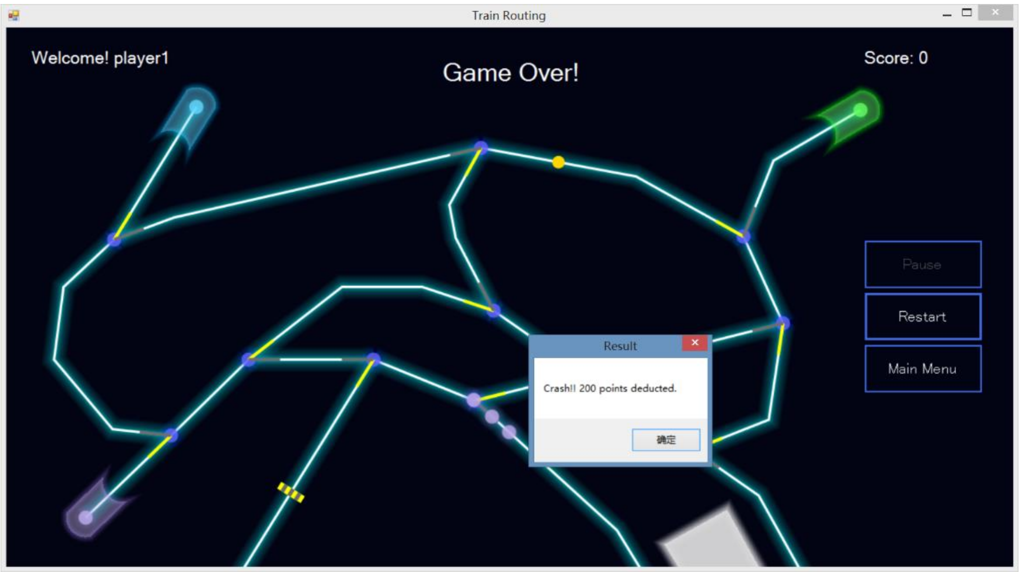
2. Two trains crash. (500 scores deducted)



Result

Crash!! 500 points deducted.

3. One train crash into a joint. (200 scores deducted)



Game Over!

Crash!! 200 points deducted.

Conclusion and further development:

Through nearly 2 months of work, we finally completed the game and achieve our objectives to a large degree.

After completing this project, we think this project is a valuable experience for us in the following two aspects. In terms of programming techniques, we became more skilled in not only C++ language but also the thoughts of a game design. If we are careless in the class designing period, we may encounter huge inconvenience at latter parts. So, the design of the program framework is of vital importance for a successful programmer. In terms of teamwork, we enjoyed the effective teamwork very much due to our reasonable work distribution. However, we found that it is kind of hard job to understand partner's codes sometimes. So we learned to form a good habit to make our program clear when programming.

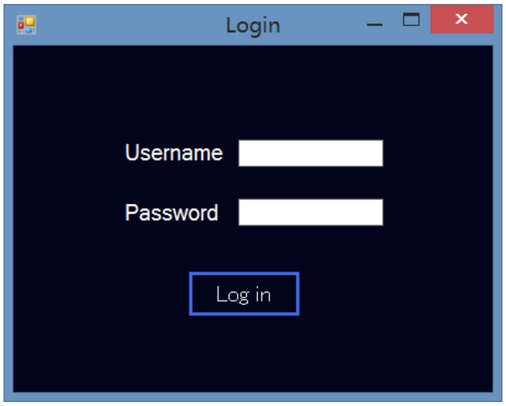
Though we finished the project on time, we think we can still optimize our program to make our game more perfect if more time is given. We plan to set a random mapping function which can arrange locations for joints randomly within limitation. This can add more uncertain factors to the game and make it more interesting. In addition, we want to add more funny modes to make the game more attractive such as Ghost Mode (all trains are ghost trains), Endless Mode (there is no time limit but the difficulty will increase with time).

Appendix

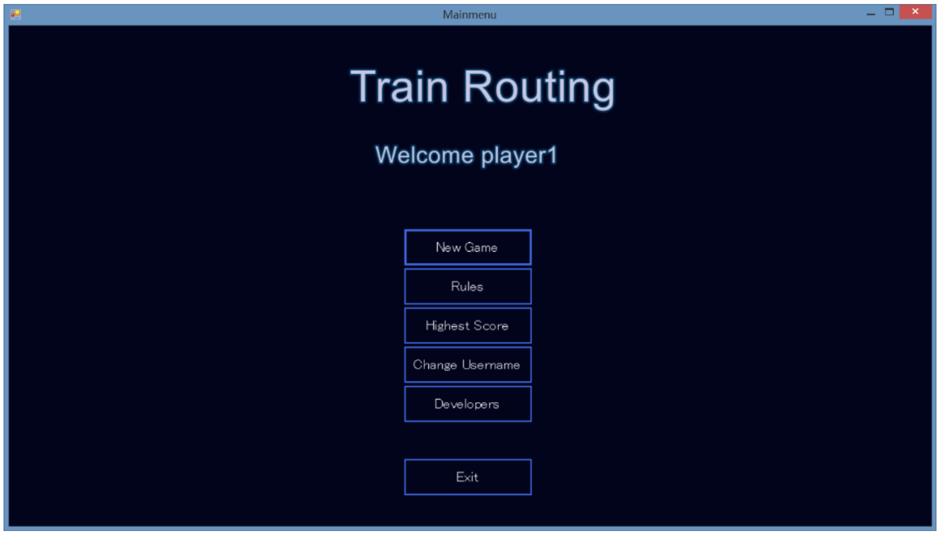
User Manuel

Operation guideline:

Upon opening the game, a Login form will be displayed. If you have played the game before, please input your username and password. If this is the first time to play this game, you can create a new account by input a username and a password and click Login.



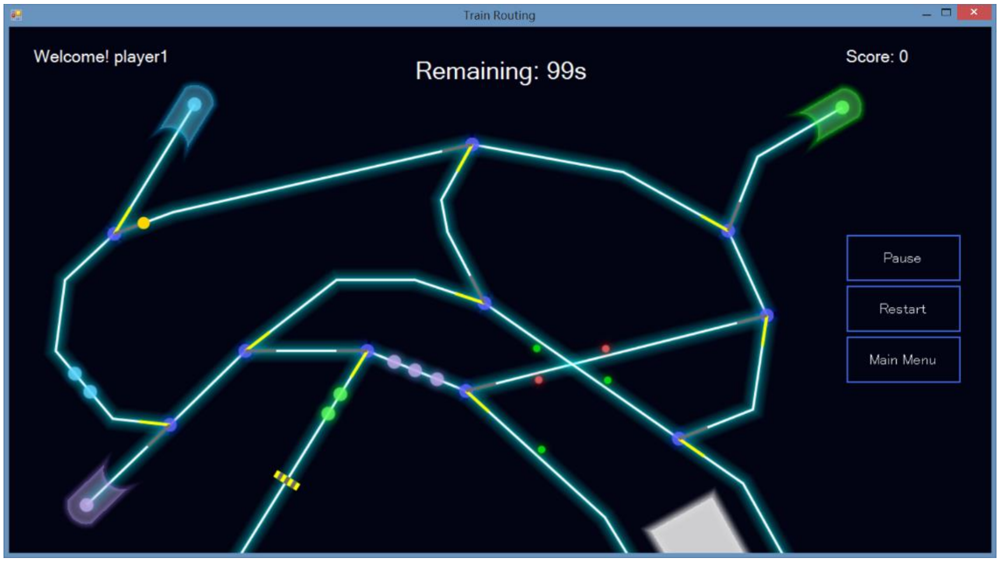
After logging in, the main menu will be shown.



- New Game: you can start a new game after choosing the difficulty.
- Rules: you will see a brief description of the game rules.
- Highest Score: your previous highest score will be shown.

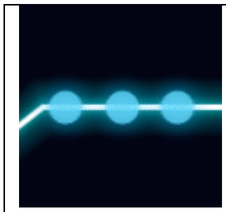
- ➢ Change Username: you can log out your present account and log in with a new account.
- ➢ Developers: you can see the information of the developers.

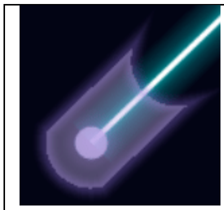
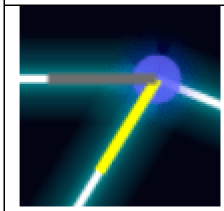
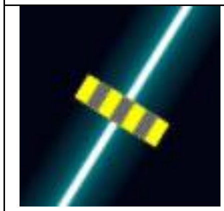
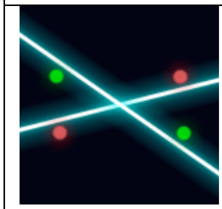
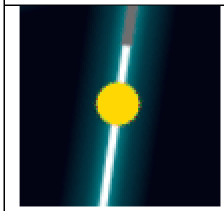
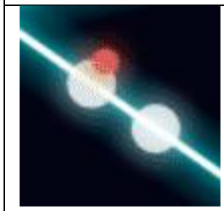
Before the game start, there will be 3 seconds freezing time. After that, you can control your joints and route for your trains!



- ➢ Pause/Resume: pause or resume the game.
- ➢ Restart: quit this game and start a new game.
- ➢ Main Menu: quit this game and go back to main menu.

Game objects instruction:

	Train: Trains will randomly come from one of the tracks of the right-bottom station with different destinations marked by different colors. You are supposed to make sure they can get their destination! But, never can you make them crashed!

	Station: A destination station accepts all trains with its color. And it will give you a 100 scores! However, it will deduct you 200 scores if you give it a wrong train.
	Joints: Joints can let trains run over them if trains come from bright tracks, while they will destroy the train when the train comes from the dark side! But they can be very tractable to change their states if you click them.
	Roadblock: Roadblock swears that he will never stop bouncing trains back from passing through it. Actually he does so.
	Traffic light: A traffic light is a hero because he can stop the train to avoid a crash if he appears red. He can turn into green and let trains pass through him if you click him.
	Bonus item: It is the most valuable thing in the world. If you get it by accident, you will earn 100 more scores!
	Ghost train: This is the most mysterious thing that you may seldom see it. It belongs to no destination station! But it will become another train with colors after a while.