PROJECT REPORT:

CS 154 : ABSTRACTIONS AND PARADIGMS PROJECT

By BIJOY SINGH KOCHAR & ANANT GUPTA

OUR PROJECT: *What you can expect…*

Our project for the course is a scheme recreation of the famous platformer Super Mario Bros. Our game covers nearly all the features that the original game provides, from power-ups, coins, bricks, maps, lives, world, Bowser and much more to explore… not to forget our own new features of AI creatures which we try to add, smart bricks, besides difficulty selection, pause feature and what we call smart-canvas, all about which you will experience when you play our version of Super Mario. Play and experience it yourself.
VERSIONS: The LITE version of the game is specially made for the NSL computers and low performance PCs, it has smaller maps *(why we did so is explained later)*.
The normal version of the game is much intensive and the map design is better and larger.

THE OBJECTIVE: *What are you supposed to do…*

The game consists of a number of worlds, each containing certain levels. A level is nothing but a map containing obstacles and active components. The objective of the game is to complete each level within a specified time interval, without losing all your lives, while gaining points and coins. Each time you die in a particular level or you run out of time, you lose one life, and you have to play that level all over again. You will be provided with five lives in the beginning of the game, and an extra life will be gained as you play through this challenge and gain 100 coins. You can replay the game at various difficulty levels.
In short, kill the creatures by jumping on them from the top or hitting the brick they stand on, obtain power-ups, gain coins, destroy bricks and challenge the time. Finally you face the Boss- Bowser, penetrate his defense and destroy him to save Princess Peach.

THE PROGRAM DESIGN: *How it works…*

- The External Structure:
  There are 3 files
  - ✓ *mainExecute.rkt* – The core propagating file containing the graphics details and loop.
  - ✓ *functions.ss* – The file containing all the classes and functions that are required by the gameplay, providing the entire engine and physics.
  - ✓ *maps.ss* – The file that helps design the maps without changing the other files.
- The Internal Structure:
  - ✓ **Classes** – Mario% : Providing the entire physics package (gravity, collision, coefficient of restitution, drag) to all active components of the game.
    – coin% : Provides the necessary functions to perform the actions of bricks springing on collision, breaking, oscillating and sensing Mario and start moving.

- ✓ **Abstractions**–In order to reduce code repetition, we've used only two classes, which can be interpreted as a large number of objects.

  Universal Class and functions: The class Mario% has been used as Mario, alien-squishy, alien-turtle, power-up and bullet (active components). By doing so, we have been able to define all collisions in terms of a single 'collision' function defined in the Mario class. The class coin% has been used to define all types of bricks, ground, flower and coins (static or dynamic components).

  Map creation: Another abstraction can be seen in the mechanism provided for designing maps. Using 'initialiser' and 'coin%-maker' functions, we have reduced map-designing to merely creating lists of the objects we need to use (i.e. coins, aliens, bricks etc.).

  Block and Point Abstraction: All the component in the game are treated as boxes of varying dimensions while they collide with each other, while physical attributes like gravity, velocity, position behave as if a point object. Benefit: A single collision function serves all the objects.

  Use of profiler: To estimate the reason of slow speeds on NSL PCs we decided to make profilers and find the root cause behind this. We concluded that the collision function is called extremely large number of times: "163397 col/sec" is one of the data. By introducing profiling and locating the problem, we understood that creating smaller maps is the key to faster speeds in slower PCs. This is seen by the fact that the collisions were now of the order of "11732 col/sec", a near 14 times improvement. *(*values are from data that was taken, it obviously varies according to gameplay)*

- ✓ **Limitations (and implemented work-around)**-

- Attaining very high speeds: Owing to the basic limitation that computations and state changes occur after finite time interval, going to very high speeds collisions can lead to unwanted results like passing through bricks. To increase the game realism and remove this bug, we give the Mario a drag and a critical speed beyond which it will not go.
- The Smart-Canvas (explained below) cannot deal with extremely low processing speeds, due to fluctuations of processing rate of the computer's processor.

- ✓ **The Smart Canvas –** The game runs with an additional feature which is extensively used in professional level games, but we tried to design it ourselves to suit are program. The game runs with a pretty much constant fps (frames per second), and an common fps on all PCs *(if is fast enough to even reach the fps)* by modulating the game's frame delay at each iteration. The fps is stored in the variable fps_data which can be called after you quit. Or you can call (avgfps) which will tell you the average fps. The default fps value is 60 (to match the screen refresh rate 60Hz).

- ✓ **The AI:**
  - ***THE TURTLES:*** The turtles and squishy aliens have been designed to sense you and follow you as you approach or flee from them. Not just that they blindly follow you, they can jump barriers and once close enough if they sense you stalking on top to jump on them, they will keep moving below you so you don't get a clear shot, when above they will try to fall off to get you. The AI level increases as difficulty increases.
  - ***THE BOSS:*** Lot smarter than the turtles the Boss will try its best to keep you away, while it sends hordes of turtle towards you, faster the closer you get. The powerful stomp will try to through you away and into spikes if you try coming close without a plan. Smart enough to dodge a bullet, shooting at him unless prepared for a quake might be a mistake. The difficulty will affect the boss's vulnerability, sight and defense.

THE FEATURES: *What all you can explore…*

GAME LAYOUT FEATURES	GAME ENGINE FEATURES	GAME MAP FEATURES
Constant fps	*Gravity*	*Breakable Bricks*
Dynamic Map	*Jump*	*Brick Bounce*
Map Generation	*Collisions*	*Power-ups*
Cleanup of Lists	*Time Clock*	*Coin Collection*
Pause	*Drag*	*Evolution*
Difficulty	*Critical Velocity*	*Automated Bricks*
Quit	*Score*	*Alien AI – (near, direction, oscillation, jump)*
Quit to Main Menu	*Lives*	*Flower*
State Completion Score	*Coins*	*Directed Shooting and Bullet Trajectory*
Restart	*Falling Off*	*Killing from below*
Fps printer	*Flag Height Score*	*Auto start platforms*
	Coefficient Of Restitution	*The third dimension*
	Auto-Accelerate	*Invisible Bricks*
		Bouncy Bricks
		Jumping Aliens
		Boss AI

ACKNOWLEDGEMENT: *Thanks to everybody who contributed to this endeavor…*

- ✓ Our Professor : ***Prof. Amitabha A. Sanyal***
- ✓ Our Dedicated T.A.s :

Gaurav Bhagwat	Ankit Gupta
Hitesh Kewalramani	Abhimanyu Khosla
Naman Mishra	Alex Poovathingal
Shruti Sharma	Anuj Kiritbhai Thakkar

(We're sorry if we missed anybody)

- ✓ http://images.google.com and lots of links (which we cannot all mention) on it which provided the necessary sprites to make the game look as you see it now.
- ✓ Nintendo for making this revolutionary game, *(Note: All the images, logo and Mario are copyright and trademark of Nintendo).*
- ✓ http://themushroomkingdom.net and http://soundbible.com for the sound effects, you can hear.
- ✓ http://www.lunapic.com for helping make images background free.
- ✓ The unspoken savior Racket Documentation.