Game Based Carrom Tutor

Mayur Katke [123050069]
Mrinal Malick [123050064]

Under the guidance of Prof. Sridhar Iyer
IIT Bombay

June 19, 2014
Introduction

- **Tutors**
  - Instructor who gives private lessons
  - Provides expertise, experience and inspiration for learning

- **Need for Carrom Tutor**
  - Very famous game
  - Many Carrom games available but not a single Carrom Tutor.

- **Carrom Tutor 1.0**

- **Carrom tutor 2.0**
Carrom

- Carrom Skills
  - Basic
  - Intermediate
  - Advanced
- Carrom strategies
  - Singles game
  - Doubles game
- Teaching Carrom
  - Expert assistance needed
  - Methods used to teach other similar games
- Carrom Games
Game based learning

- Practice versus theory
- Properties of GBL
  - Interactivity
  - Motivation for learners
  - Curiosity driver
- Steps for building educational game
  - What?
  - Why?
  - How?
- Assessment
- Tutoring
Carrom Tutor 1.0

Design

- ET perspective
  - Scaffolding
  - Sequencing
  - Cognitive model of mind
  - Recall level exercises

- Design
  - Demonstration of skills with text explanation
  - Exercises for testing user
Carrom Tutor 1.0
Implementation

- **HTML**
  - Webpages in tutor were created using HTML

- **CSS**
  - Used for overall designing of webpages
  - File storing all CSS properties has been included in all html pages

- **JavaScript**
  - Used for taking user inputs
  - Changes the content of webpages according to inputs
  - Evaluation and tutoring was done in JavaScript

- **Macromedia Flash MX**
  - Used for creating all animated *gifs* used in tutor
Carrom Tutor 1.0

User Interface

Figure: Exercise page of Carrom Tutor 1.0
User Experiments

- Learning gain
  - How did system helped in learning Carrom?
  - How many new carrom skills have you learned?
  - Was the tutor interactive?

- Usability of the tutor
  - SUS(System Usability Scale) analysis was done
  - Five point likert questions are asked in SUS form
User Experiments

Carrom Tutor 1.0

Average percentage of SUS analysis is 77.14. Average responses for each question asked in SUS analysis are plotted in following graph.

Figure: SUS feedback for Carrom Tutor 1.0
Motivation for Carrom Tutor 2.0

Shortcomings of Carrom Tutor 1.0

- Pre-decided places for striker
- Users thinking restricted
- Few pre-decided shots were displayed to user
- Less flexibility and control
- Better game like environment can be provided
Design of Carrom Tutor 2.0

- **ET perspective**
  - Modelling
  - Sequencing
  - Cognitive model of mind

- **Design perspective**
  - Practice exercises after demonstration for each skill
  - Placing striker anywhere on baseline in exercises
  - User can see whatever shot she plays
  - Force gauge for deciding force on striker
  - 3D game environment provided to users
Blender

- Desired features and functionalities can be provided in game like environment.
- To make it seem like real playing experience 3D interface can be provided.
- Game engine should be used to provide such environment.
- Blender is very popular and open source Game Engine used for creating games, animations, object models etc.
- Blender game engine was used for building Carrom Tutor 2.0.
Implementation

There are four main parts of implementation in Blender Game Engine.

- Modelling
  - Objects, characters, and scenes are created using modelling.
  - Many shapes are available for it.

- Animation
  - Animations were created in Blender Render using timeline feature.
  - Outputs of these animations were taken as image sequences.
  - Videos used in Carrom Tutor 2.0 were created from these image sequences.

- Logic Editor

- Python Scripting
Logic Editor

Blender Game Engine provides a scripting layer called Game Logic. It has three main parts

- Logic bricks
- Properties
- States

Logic brick is most important part of Game Logic. It has three main components.

- Sensors
- Controllers
- Actuators
Logic Editor

**Figure:** Basic scene

- Basic
- Intermediate
- Advanced

- Straight shot
- Normal cut
- Straight cut
- Negative cut
- Doubling

hit the coin in large angle towards pocket
Logic Editor

Figure: Logic bricks for basicScene
Logic Editor

Figure: Properties used in the screen used in basicScene
**Figure:** Logic bricks for playing video and loading practice exercise
Logic Editor

**Figure:** Logic editor for a coin
Logic Editor

Saving after a black coin goes into pocket

Figure: Saving and loading
Logic Editor

- There are total 46 game scenes.
- Most of these scenes have very complex interconnections between different game objects.
Logic Editor

Figure: Basic scene’s logic bricks, part 1
Figure: Basic scene’s logic bricks, part 2
Logic Editor

Figure: Basic scene’s logic bricks, part 3
Python Scripting

- Python scripting language in blender provides
  - Special interfaces to access blender’s internal functions
  - Ability to extend functionality of system
- Python API is integrated with blender.
- This API can manipulate any object properties in game.
- Python scripts can be used in controllers
Python Scripts in Carrom Tutor 2.0

- There are total 65 python scripts written for Carrom Tutor 2.0.
- In each scene there are six main scripts applied on different objects
  - Striker.py
  - Score.py
  - EmptyMove.py
  - RaySensor.py
  - Mousemove.py
  - Movie.py
```
# Python Script

lKey = bge.logic.KX_INPUT_ACTIVE ==
    keyboard.events[bge.events.LEFTARROWKEY]

rKey = bge.logic.KX_INPUT_ACTIVE ==
    keyboard.events[bge.events.RIGHTARROWKEY]

lClick = bge.logic.KX_INPUT JUST_ACTIVATED ==
    mouse.events[bge.events.LEFTMOUSE]

if not striker["slider_enabled"]:
    if (lKey and striker.position[0] > -21.62116):
        striker.applyMovement((-0.1, 0, 0), False)
    if (rKey and striker.position[0] < 22.45835):
        striker.applyMovement((0.1, 0, 0), False)

if lClick:
    if ((stricker.position[0] <= -18.31307 and
        striker.position[0] >= -20.75888) or
        (stricker.position[0] <= 21.53248 and
        striker.position[0] >= 18.84937)):
        print("Wrong striker placing")
    else:
        striker["slider_enabled"] = True

lClick = 0
```

**Figure:** Move striker and check half ball position
Python Script

```python
if ((striker["slider_enabled"]) and (not striker["striker_released"])):
    half_time_period = 70
    time_period = 2 * half_time_period

    striker["timer"] += 1
    if (striker["timer"] -- time_period):
        striker["timer"] = 1
        v = (sin(3.141592*(striker["timer"])/half_time_period) - 1.570796)+1)*500

    if (v > 0):
        striker["slider_1"] = True
        striker["force_multiplier"] = 5
    else:
        striker["slider_1"] = False

    if (v > 100):
        striker["slider_2"] = True
        striker["force_multiplier"] = 10
    else:
        striker["slider_2"] = False

    if (v > 200):
        striker["slider_3"] = True
        striker["force_multiplier"] = 15
    else:
        striker["slider_3"] = False
```

**Figure:** Power slider characteristics

Applying sine-wave characteristics to slider

Applying force multiplier according to slider height
```python
force = 0

if not striker["striker_released"] and lClick:
    vector_magnitude_x = math.sqrt((camera.worldOrientation[0][0] ** 2) + (camera.worldOrientation[1][0] ** 2))
    vector_magnitude_y = math.sqrt((camera.worldOrientation[0][1] ** 2) + (camera.worldOrientation[1][1] ** 2))

    striker.worldOrientation[0][0] = camera.worldOrientation[0][0] / vector_magnitude_x
    striker.worldOrientation[0][1] = camera.worldOrientation[0][1] / vector_magnitude_x
    striker.worldOrientation[2][0] = 0
    striker.worldOrientation[0][1] = camera.worldOrientation[0][1] / vector_magnitude_y

    striker.worldOrientation[1][1] = camera.worldOrientation[1][1] / vector_magnitude_y
    striker.worldOrientation[2][1] = 0

    force = striker["force_multiplier"] * 90
    striker.applyForce((0, force, 0), True)
    striker["striker_released"] = True
```

**Figure:** Striker’s rotation according to camera movement
Figure: Updating score according to user’s shot
Work distribution

- **Modelling**
- **Logic editor**
  - Logic bricks related to save/load, detecting coin pocketing were implemented by Mayur
  - Logic bricks related to UI designing, displaying video and scene manipulation were implemented by Mrinal
- **Python scripting**
  - Striker movement, force gauge, empty movement, mouse movement were mainly implemented/used by Mayur
  - Ray sensor, scoring, coin detection on baseline, display of movie were implemented/used by Mrinal

Practice and complex exercises were equally distributed for implementation
Sample

- Target audience
- Mixture of novice, intermediate and experts
- Eleven Users took part in experiment of Carrom Tutor 2.0
Data Collection Methodology

- Pre and post tests were conducted
- Usability
  - SUS analysis
  - Five point likert questions
  - Check accessibility, efficiency, effectiveness, attractiveness
- Learning gain
  - Users were asked some questions and they filled one form
  - Comparison with other applications was done.
SUS analysis

Following questions are asked to users in SUS analysis form.

1. I think that I would like to use the system frequently
2. I found the system unnecessarily complex
3. I thought the system was easy to use
4. I think I need the support of a technical person to use the system
5. I found various functions of the system were well integrated
6. I thought there was too much inconsistency in the system
7. I would imagine most people will learn to use the system very quickly
8. I found the system very cumbersome to use
9. I felt very confident using the system
10. I needed to learn a lot of things before I could get going with the system
SUS analysis was done for Carrom Tutor 2.0 and in addition users were asked to fill one form. Questions to check learning gain of users were asked in this form. Average percentage of SUS score is 84.09.

![SUS feedback for Carrom Tutor 2.0](image_url)
User Experiments
SUS

- Answers to the SUS questions were in range from strongly disagree to strongly agree
- A SUS score of 68 is generally considered as average
- Carrom Tutor 1.0 has an average score of 77.14 (grade B)
- Carrom Tutor 2.0 has an average score of 84.09 (grade A)
Questions were asked to users for checking the learning gain in feedback form.

- User’s exposure to Carrom
- How many new skills you learned?
- How was it playing the exercises before watching demos?
- How was it playing the exercises after watching the demos?
- How difficult was it to relate the demos with the shots needed to play?
- Compare with other Carrom application you played with
References

References

Thank You.