Deakin University

Object Oriented Development

OnTrack Submission

A Simple Reaction-Timer Controller

Submitted By:
Peter Stacey
pstacey
2020/05/10 18:12

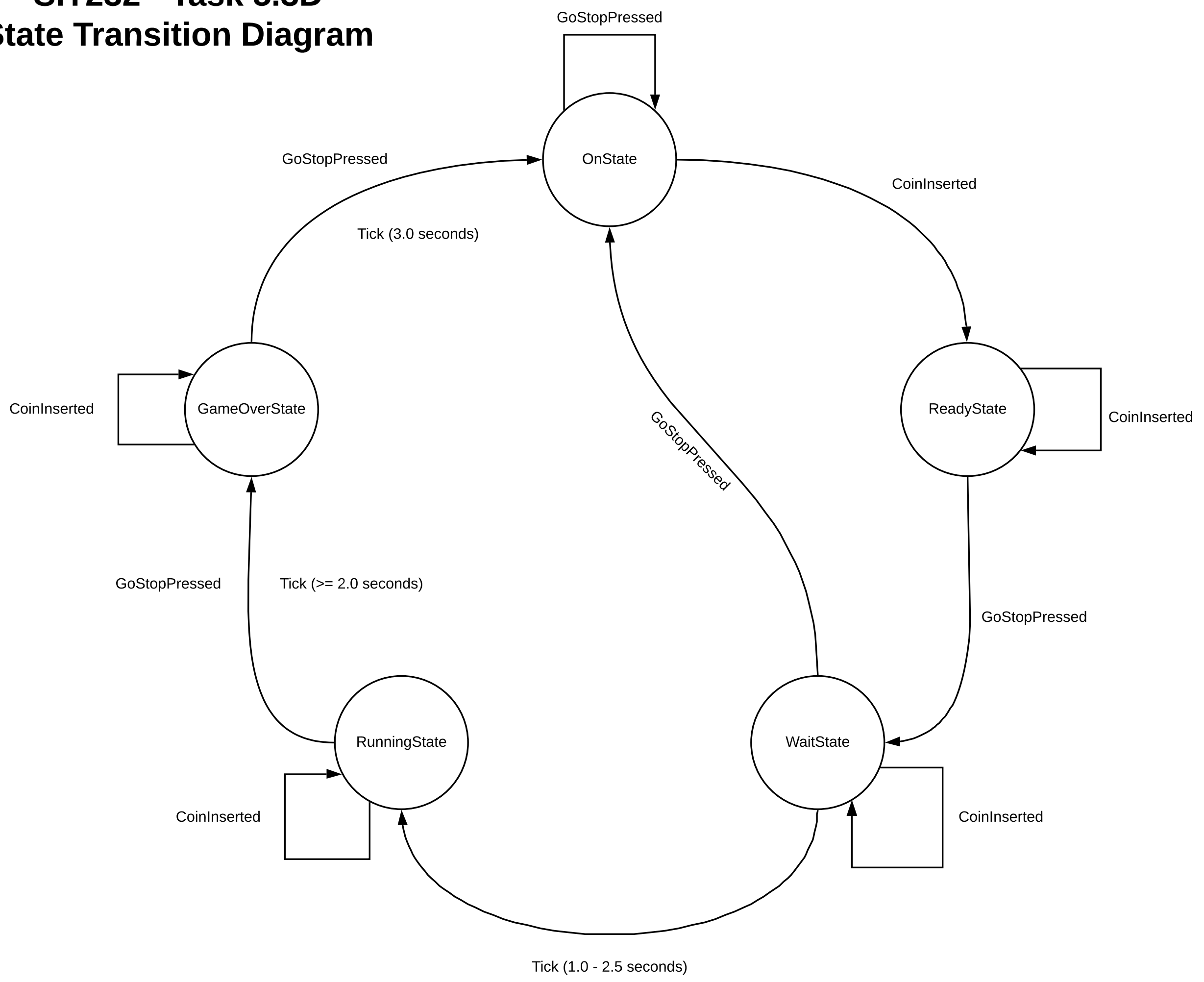
Tutor:
Dipto Pratyaksa

Outcome	**Weight**
Evaluate Code	♦♦♦♦◊
Principles	♦♦♦◊◊
Build Programs	♦♦♦♦◊
Design	♦♦♦♦♦
Justify	♦♦♦◊◊

By providing the base files, this task involved evaluating an existing program in order to understand how to integrate our additional file into it. It also involved applying the principle of OOP, to encapsulate data, abstract the design our to a recognised OOP design pattern, to implement the design through coding the required file using a State pattern and supporting our design with a diagram of the states and transitions. With the transition diagram and file, we have evidence of our effort and this is further supported by my video.

May 10, 2020

SIT232 - Task 5.3D
State Transition Diagram



```csharp
using SimpleReactionMachine;
using System;
using System.Data;

namespace SimpleReactionMachine
{
    public class SimpleReactionController : IController
    {
        // Settings for the game times
        private const int MIN_WAIT_TIME = 100; // Minimum wait time, 1 sec in ticks
        private const int MAX_WAIT_TIME = 250; // Maximum wait time, 2.5 sec in
         ↪  ticks
        private const int MAX_GAME_TIME = 200; // Maximum of 2 sec to react, in
         ↪  ticks
        private const int GAMEOVER_TIME = 300; // Display result for 3 sec, in ticks
        private const double TICKS_PER_SECOND = 100.0; // Based on 10ms ticks

        // Instance variables and properties
        private State _state;
        private IGui Gui { get; set; }
        private IRandom Rng { get; set; }
        private int Ticks { get; set; }

        /// <summary>
        /// Connects the controller to the Gui and Random Number Generator
        /// </summary>
        /// <param name="gui">IGui concrete implementation</param>
        /// <param name="rng">IRandom concreate implementation</param>
        public void Connect(IGui gui, IRandom rng)
        {
            Gui = gui;
            Rng = rng;
            Init();
        }

        /// <summary>
        /// Initialises the state of the controller at the start of the program
        /// </summary>
        public void Init()
        {
            _state = new OnState(this);
        }

        /// <summary>
        /// Coin inserted event handler
        /// </summary>
        public void CoinInserted()
        {
            _state.CoinInserted();
        }

        /// <summary>
        /// Go/Stop pressed event handler
        /// </summary>
        public void GoStopPressed()
        {
            _state.GoStopPressed();
        }

        /// <summary>
        /// Tick event handler
        /// </summary>
        public void Tick()
        {
            _state.Tick();
        }

        /// <summary>
        /// Sets the state of the controller to the desired state
        /// </summary>
        /// <param name="state">The new state to transition to</param>
        private void SetState(State state)
        {
            _state = state;
        }

        /// <summary>
        /// Base class for concrete State classes
        /// </summary>
        private abstract class State
        {
            protected SimpleReactionController _controller;

            public State(SimpleReactionController controller)
            {
                _controller = controller;
            }

            public abstract void CoinInserted();
            public abstract void GoStopPressed();
            public abstract void Tick();
        }

        /// <summary>
        /// State of the game when it is waiting for a coin to be inserted
        /// </summary>
        private class OnState : State
        {
            public OnState(SimpleReactionController controller) : base(controller)
            {
                _controller.Gui.SetDisplay("Insert coin");
            }

            public override void CoinInserted()
            {
                _controller.SetState(new ReadyState(_controller));
            }
            public override void GoStopPressed() { }
            public override void Tick() { }
        }

        /// <summary>
        /// State of the game when a coin has been inserted, but the game is not yet
        /// started
        /// </summary>
        private class ReadyState : State
        {
            public ReadyState(SimpleReactionController controller) :
             ↪  base(controller)
            {
                _controller.Gui.SetDisplay("Press Go!");
            }

            public override void CoinInserted() { }
            public override void GoStopPressed()
            {
                _controller.SetState(new WaitState(_controller));
            }
            public override void Tick() { }
        }

        /// <summary>
        /// State of the game when the game has started and it is waiting for the
        /// random time
        /// </summary>
        private class WaitState : State
        {
            private int _waitTime;
            public WaitState(SimpleReactionController controller) : base(controller)
            {
                _controller.Gui.SetDisplay("Wait...");
                _controller.Ticks = 0;
                _waitTime = _controller.Rng.GetRandom(MIN_WAIT_TIME, MAX_WAIT_TIME);
            }

            public override void CoinInserted() { }
            public override void GoStopPressed()
            {
                _controller.SetState(new OnState(_controller));
            }
            public override void Tick()
            {
                _controller.Ticks++;
                if(_controller.Ticks == _waitTime)
                {
                    _controller.SetState(new RunningState(_controller));
                }
            }
        }
```

```csharp

        /// <summary>
        /// State of the game when the timer is counting and it is waiting for the
        /// user to react by pressing the Go/Stop button
        /// </summary>
        private class RunningState : State
        {
            public RunningState(SimpleReactionController controller) :
             ↪  base(controller)
            {
                _controller.Gui.SetDisplay("0.00");
                _controller.Ticks = 0;
            }

            public override void CoinInserted() { }
            public override void GoStopPressed()
            {
                _controller.SetState(new GameOverState(_controller));
            }

            public override void Tick()
            {
                _controller.Ticks++;
                _controller.Gui.SetDisplay(
                    (_controller.Ticks / TICKS_PER_SECOND).ToString("0.00"));
                if(_controller.Ticks == MAX_GAME_TIME)
                {
                    _controller.SetState(new GameOverState(_controller));
                }
            }
        }

        /// <summary>
        /// State of the game when the time has expired, or the user reacted.
        /// </summary>
        private class GameOverState : State
        {
            public GameOverState(SimpleReactionController controller) :
             ↪  base(controller)
            {
                _controller.Ticks = 0;
            }

            public override void CoinInserted() { }
            public override void GoStopPressed()
            {
                _controller.SetState(new OnState(_controller));
            }
            public override void Tick()
            {
                _controller.Ticks++;
                if(_controller.Ticks == GAMEOVER_TIME)
                {
                    _controller.SetState(new OnState(_controller));
                }
            }
        }
    }
}
```