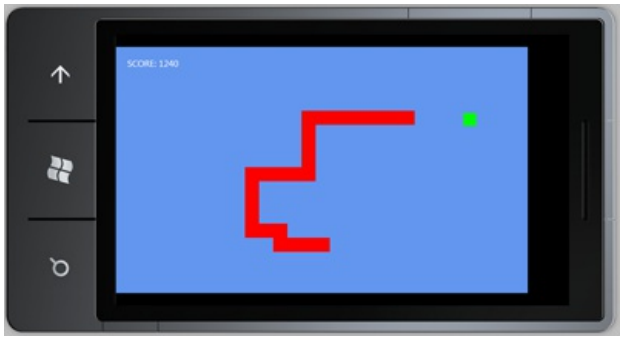
Snake game in Windows Phone 7

This article explains how to create a simple snake game in Windows Phone 7.

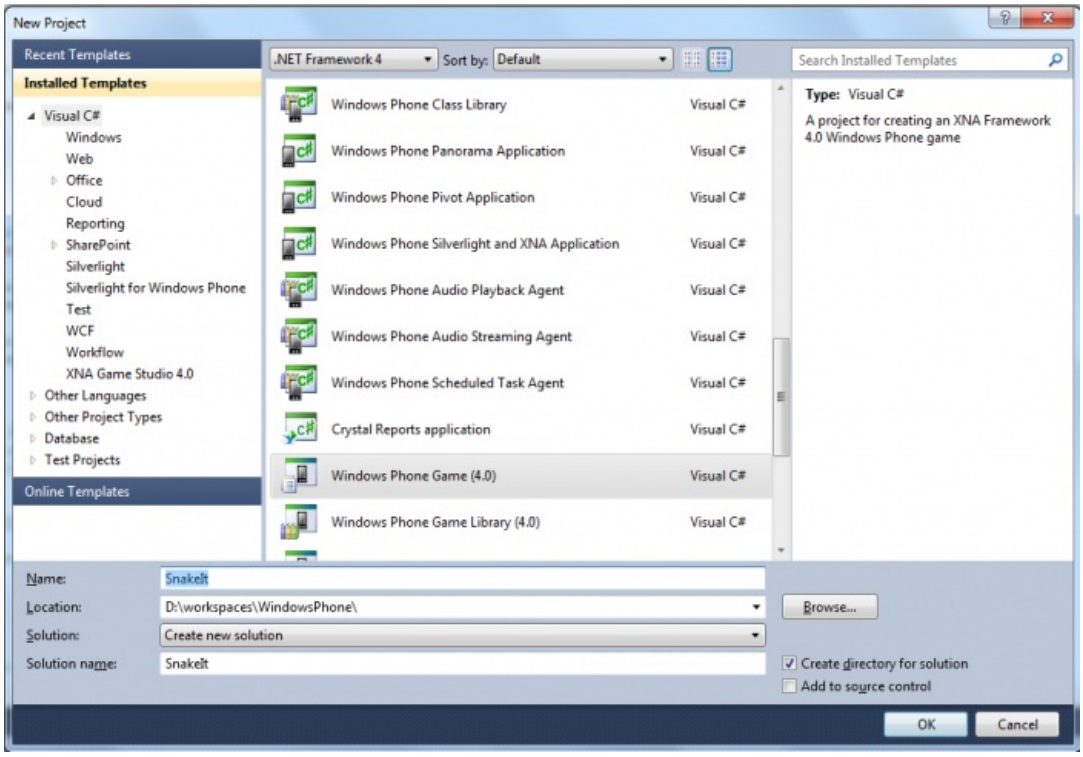
Introduction

In this article, we will show how to create a simple Snake Game in Windows Phone. You can move snake in different directions with gestures. The game contains simple Menu, Game and Complete Screens.



Windows Phone Game

To start creating a new Windows Phone Game, start Microsoft Visual studio then create a new Project and select Windows Phone Game (4.0) Template. We use SnakeIt as a project name in this example.



In this example, we have used C# as code behind language.

Game Graphics

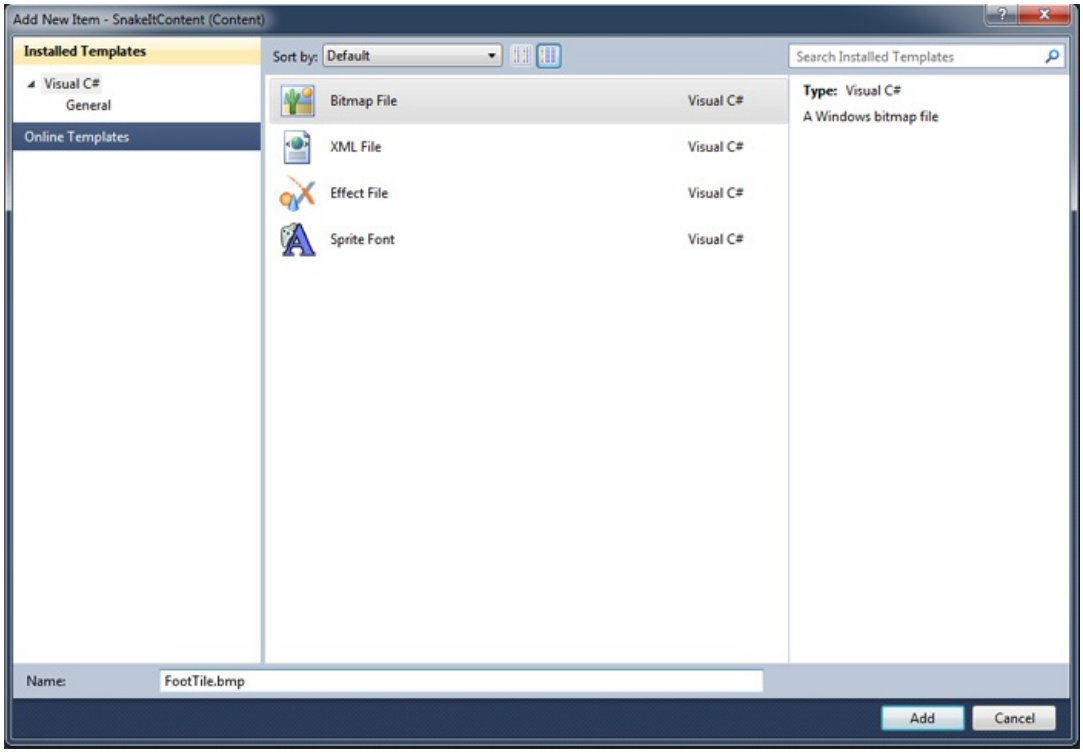
(C) Copyright Nokia 2014. All rights reserved.

First we have to add basic graphics for the food and snake. In this example we have used 25x25 tile based graphics.

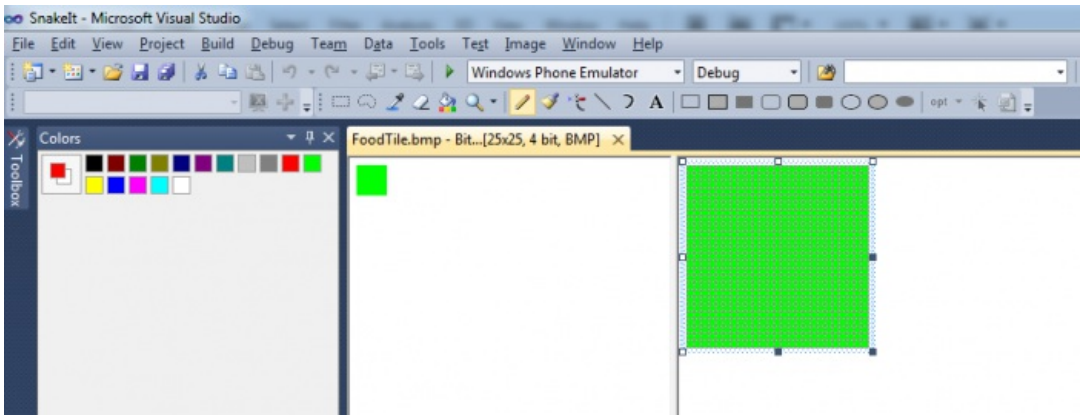
Food

Right click SnakeIt project Content in Solution Explorer and select Add > New Item...

Select Bitmap File and name it to FootTile.bmp



Draw 25x25 green rectangle as a food graphics. Remember save your graphics.



Snake

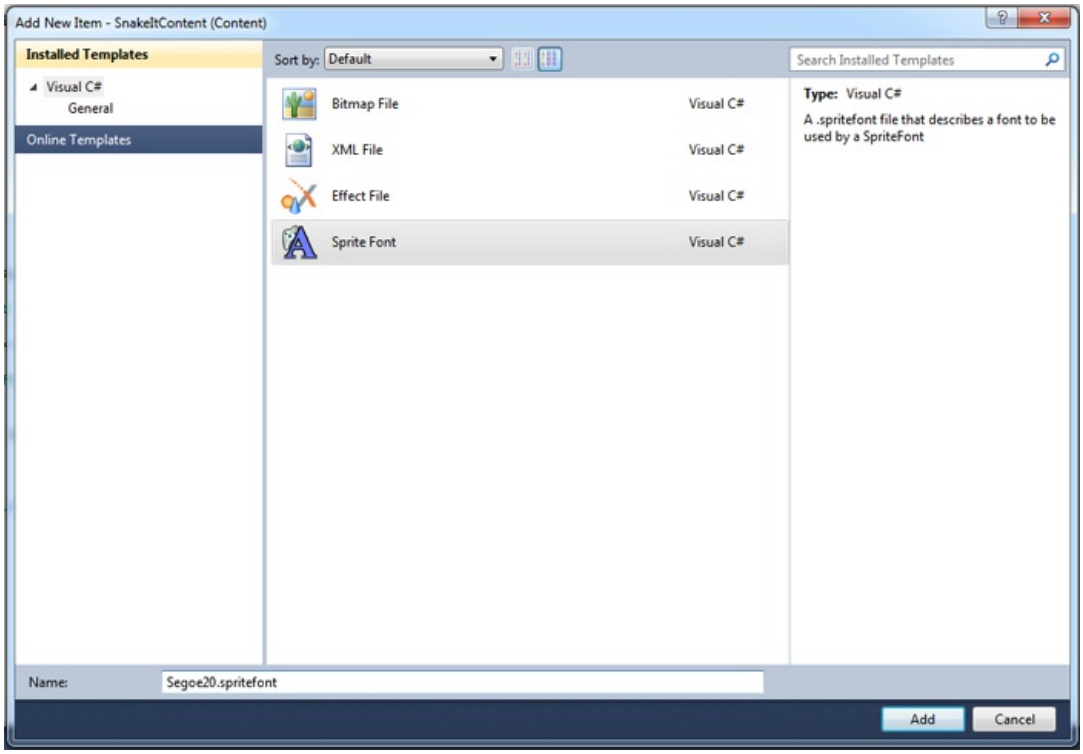
Do the same thing with Snake Graphic and use red color (for example). Name it to SnakeTile.bmp.

Game Font

We can create a new font to the game same way as we did the graphics. Right click your project content in Solution Explorer and select Add > New Item...

Select Sprite Font and name it to Segoe20.spritefont

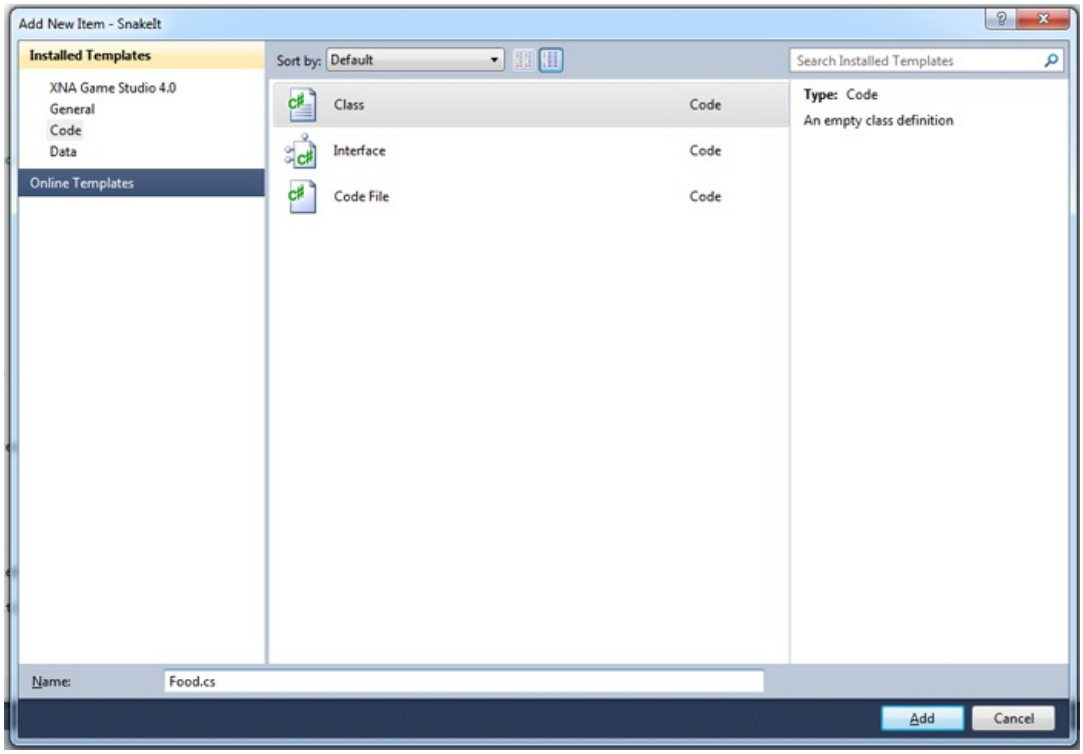
(C) Copyright Nokia 2014. All rights reserved.



Double click font in Solution Explorer and change font size to 20. Remember to save your font.

Creating a Food Class

Right click your project from Solution Explorer and select Add > New Item... Select Code from Installed Templates and create a Food Class.



Food Class stores food's `Texture` and `Position`. Food's `width` and `height` is used in collision detection from `Game1` Class.

```
using System;
using Microsoft.Xna.Framework;
using Microsoft.Xna.Framework.Graphics;

namespace SnakeIt
{
    class Food
```


(C) Copyright Nokia 2014. All rights reserved.

```
    {
        // Texture representing the food
        public Texture2D Texture;

        // Position of the food relative to the upper left side of the screen
        public Vector2 Position;

        // Get the width of the food
        public int Width
        {
            get { return Texture.Width; }
        }

        // Get the height of the food
        public int Height
        {
            get { return Texture.Height; }
        }

        // Set Food Texture and Position
        public void Initialize(Texture2D texture, Vector2 position)
        {
            Texture = texture;
            Position = position;
        }

        // Draw Food to the Screen
        public void Draw(SpriteBatch spriteBatch)
        {
            spriteBatch.Draw(Texture, Position, null, Color.White, 0f, Vector2.Zero, 1f,
SpriteEffects.None, 0f);
        }
    }
}
```

Creating a Snake Class

Create a Snake Class same way as Food is described in the previous section.

Usings

We use XNA Framework and List from `Collections` to handle all the tiles of the Snake.

```
using System;
using Microsoft.Xna.Framework;
using Microsoft.Xna.Framework.Graphics;
using System.Collections.Generic;
```

Properties

A few properties are used to handle the Snake. We will load our Snake Graphics to Texture and set the snake initial position to middle of the screen (in Game1 Class later). All the snake tiles will be store to `snakeTiles` list. Snake direction will be handled from `Game1` class with gestures.

```
// Snake texture
public Texture2D Texture;
```


(C) Copyright Nokia 2014. All rights reserved.

```
// Snake initial position
public Vector2 Position;

// Snake Tiles
public List<Tile> snakeTiles;

// Snake speed and increment
public float speed;
const float speedIncrement = 0.2f;

// Snake direction // 1==left, 2==up, 3==right, 4==down
public int direction;

// is collision detected in Game -> add a new tile to the Snake
public bool newTile;
```

Initialize

`Game1` will initialize the Snake object. Snake's texture will be loaded and the initial position is set. Snake's properties will be reset in first run and later when player want's to play game again. Snake moves 5 pixels to left by default and the head of the snake will be created.

```
// initialize Snake
 public void Initialize(Texture2D texture, Vector2 position)
{
       Position = position;
       Texture = texture;
       Reset();
}

// reset snake (a new game)
public void Reset()
{
      speed = 5f;
      direction = 1;
      Tile tile = new Tile(Texture, Position);
      snakeTiles = new List<Tile>();
      snakeTiles.Add(tile);
      newTile = false;
}
```

Update

The game will call to update Snake in every game step. Here we first find the head of the Snake (it is the first tile of the `snakeTiles` List). If we had to add a new tile to the snake (food is eaten), then we create a new tile to `headPostion` and insert it to `snakeTiles` List. Next we move only the head of the snake and this way a snake is bigger. If there is no new tile coming to the snake, we only move last tile to snake's "head" and move it to a new position.

```
public void Update()
{
            // Get Snakes Head Position
            Vector2 headPosition = snakeTiles[0].Position;

            // add a new tile
```


(C) Copyright Nokia 2014. All rights reserved.

```
            if (newTile)
            {
                // create a new tile
                Tile tile = new Tile(Texture, headPosition);
                // insert new tile to SnakeTiles (after HeadPosition, but in same
position in screen)
                snakeTiles.Insert(1, tile);
            }
            // move last snake tile to the head position in screen
            else if (snakeTiles.Count > 1)
            {
                Tile last = snakeTiles[snakeTiles.Count - 1];
                last.Position = headPosition;
                snakeTiles.RemoveAt(snakeTiles.Count-1);
                snakeTiles.Insert(0, last);
            }

            // more speed if snake has a new tile (eat)
            if (newTile)
            {
                speed += speedIncrement;
                newTile = false;
            }

            // Move Snake head tile to the new position in the screen
            switch (direction)
            {
                case 1: snakeTiles[0].Position.X -= speed; break;
                case 2: snakeTiles[0].Position.Y -= speed; break;
                case 3: snakeTiles[0].Position.X += speed; break;
                case 4: snakeTiles[0].Position.Y += speed; break;
            }
 }
```

Draw

Every tile of the Snake is drawn in `Draw()`-method. This method is called from `Game1` Class.

```
public void Draw(SpriteBatch spriteBatch)
{
            for (int i = 0; i < snakeTiles.Count; i++)
            {
                spriteBatch.Draw(snakeTiles[i].Texture, snakeTiles[i].Position, null,
Color.White, 0f, Vector2.Zero, 1f, SpriteEffects.None, 0f);
            }
}
```

Tile Class (Inner Class in Snake Class)

We have used a inner `Tile` Class in Snake. This class only define Snake tile. Every tile has `Position` and `Texture`.

```
class Tile
{
        public Vector2 Position;
        public Texture2D Texture;
```


(C) Copyright Nokia 2014. All rights reserved.

```
        public Tile(Texture2D texture, Vector2 position) {
            Position = position;
            Texture = texture;
        }

        // Get the width of the snake
        public int Width
        {
            get { return Texture.Width; }
        }

        // Get the height of the snake
        public int Height
        {
            get { return Texture.Height; }
        }
}
```

Game1 Class

The `Game1` Class is main class in Windows Phone XNA Games. Here we initialize and load Snake game contents. Gestures and collision detection is checked in game's `Update` method. `Draw` method check `GameMode` Property and draws Menu, Game or Completed information to the screen.

Usings

Normal XNA Classes are used and Touch to detect gestures.

```
using System;
using System.Linq;
using Microsoft.Xna.Framework;
using Microsoft.Xna.Framework.Graphics;
using Microsoft.Xna.Framework.Input;
using Microsoft.Xna.Framework.Input.Touch;
```

Properties

We use `GraphicsDeviceManage` Class to get information of the current Display. `SpriteBatch` Class is used to draw information to the screen. We have decided to use different game modes to control `Draw()`-method to draw menu, game or complete screens to device. This is easy way to handle drawing situations, there is component based possibility too (you should check this link: Game State Management). Current game mode is stored to `Mode` property, all the possible modes are in enum `GameMode`.

```
// Information about current display
private GraphicsDeviceManager graphics;
// Draw game objects
private SpriteBatch spriteBatch;
// The snake
private Snake snake;
// The food
private Food food;
// Sample Font
private SpriteFont segoe20;
// Score position in Screen
private Vector2 scorePosition;
```


(C) Copyright Nokia 2014. All rights reserved.

```
// Score
private float score;
private const float FOOD_POINTS = 20;
// Current Game Mode
 GameMode Mode;
// Different Game Modes (Menu, Running, Complete)
enum GameMode {Menu, Running, Complete}
```

Constructor

First we connect to `GraphicsDeviceManage` Class and tell `Content` Manager our root directory of Content (Graphics and Font). Frame rate is set to default one 30 fps. We need to enable Tap and Flick Gestures. Different GameModes are changed with Tap gestures and snake movement is handled with Flick's.

```
public Game1()
{
            // Graphics Device manager
            graphics = new GraphicsDeviceManager(this);

            // Content manager root directory
            Content.RootDirectory = "Content";

            // Frame rate is 30 fps by default for Windows Phone.
            TargetElapsedTime = TimeSpan.FromTicks(333333);

            // Extend battery life under lock.
            InactiveSleepTime = TimeSpan.FromSeconds(1);

            // Detech Tap and Flick gestures
            TouchPanel.EnabledGestures = GestureType.Tap | GestureType.Flick;
 }
```

Initialize the Game

`Initialize` method allows the game to perform any initialization it needs to before starting to run. Here we set the game mode and create Snake and Food objects.

```
protected override void Initialize()
{
            // Set Game Mode to Menu, Menu Screen will be drawn in Draw()-method
            Mode = GameMode.Menu;
            // Create Snake and Food
            snake = new Snake();
            food = new Food();
            // set Score to zero
            score = 0f;

            base.Initialize();
}
```

Loading Content

`LoadContent` will be called once per game and is the place to load all of your content. First we connect to graphics device with `SpriteBatch` so we can later draw objects to the screen. Next we load and create Textures for the Snake and Food. Snake and Food `Initialize` method will be called now. We create our font here and calculate the position of the score text too.

(C) Copyright Nokia 2014. All rights reserved.

```
protected override void LoadContent()
{
            // Create a new SpriteBatch, which can be used to draw textures.
            spriteBatch = new SpriteBatch(GraphicsDevice);

            // Create a Snake Texture and put the snake at the middle of the screen
            Vector2 snakePosition = new Vector2(
                GraphicsDevice.Viewport.TitleSafeArea.X +
GraphicsDevice.Viewport.TitleSafeArea.Width / 2,
                GraphicsDevice.Viewport.TitleSafeArea.Y +
GraphicsDevice.Viewport.TitleSafeArea.Height / 2);
            snake.Initialize(Content.Load<Texture2D>("SnakeTile"), snakePosition);

            // Create a Food Texture and put it to random position of the screen
            Vector2 foodPosition = RandPosition();
            food.Initialize(Content.Load<Texture2D>("FoodTile"), foodPosition);

            // Create a Font
            segoe20 = this.Content.Load<SpriteFont>("Segoe20");

            // Set the score text position at the top left corner of the screen
            scorePosition = new Vector2(20, 20);
}
```

Updating the Game

`Update` method allows the game to run logic such as updating the world checking for collisions, gathering input and playing audio. Here we update the snake and check collision if the game is running. Gestures will be checked in every step.

```
protected override void Update(GameTime gameTime)
{
            // Allows the game to exit
            if (GamePad.GetState(PlayerIndex.One).Buttons.Back == ButtonState.Pressed)
                this.Exit();

            if (Mode == GameMode.Running)
            {
                snake.Update();
                CheckCollision();
            }

            CheckGestures();

            base.Update(gameTime);
}
```

Checking Collision

Snake's head and food position is calculated to `Rectangle` objects. Then we can use the Rectangle's built-in intersect function to determine if those are overlapping. We give more points and set the food to a new random position if collision is detected (and we give hint to the Snake to add a new tile in next update). The game is over if the snake's head is moved from the screen area.

```
private void CheckCollision()
{
```


(C) Copyright Nokia 2014. All rights reserved.

```
            // Use the Rectangle's built-in intersect function to determine if two
objects are overlapping
            int snakeHeadX = (int)snake.snakeTiles[0].Position.X;
            int snakeHeadY = (int)snake.snakeTiles[0].Position.Y;
            Rectangle rectangle1 = new Rectangle(snakeHeadX, snakeHeadY,
snake.snakeTiles[0].Width,  snake.snakeTiles[0].Height);
            Rectangle rectangle2 = new Rectangle((int)food.Position.X,
(int)food.Position.Y, food.Width, food.Height);

            // Snake and Food
            if (rectangle1.Intersects(rectangle2))
            {
                // add score
                score += FOOD_POINTS;
                // change food position
                food.Position = RandPosition();
                // new tile to snake
                snake.newTile = true;
            }

            // Snake and Screen Boundaries
            if (snakeHeadX < 0 ||
                snakeHeadY < 0 ||
                snakeHeadX + snake.snakeTiles[0].Width > GraphicsDevice.Viewport.Width
||
                snakeHeadY + snake.snakeTiles[0].Height >
GraphicsDevice.Viewport.Height)
            {
                Mode = GameMode.Complete;
            }
}
```

Below method is used to move food to the new random position in the screen.

```
private Vector2 RandPosition()
{
            // random food to a new position
            Random random = new Random();
            Vector2 position = new Vector2(
            GraphicsDevice.Viewport.TitleSafeArea.X +
random.Next(GraphicsDevice.Viewport.TitleSafeArea.Width - 45) + 20,
            GraphicsDevice.Viewport.TitleSafeArea.Y +
random.Next(GraphicsDevice.Viewport.TitleSafeArea.Height - 45) + 20);
            return position;
}
```

Checking Gestures

Gestures will be checked in every `Update` run in the game. First we check if there are new gestures available, if yes, then we read those. Drag `Delta` amount is calculated if `Flick` type is detected. Snake direction is moved that direction which is happened. Player cannot move snake to opposite direction (Game is over). Game mode is changed if `Tap` gesture is detected (it only start the game from menu or restart it).

```
private void CheckGestures()
{
```


(C) Copyright Nokia 2014. All rights reserved.

```
            // is there gestures available
            while (TouchPanel.IsGestureAvailable)
            {
                // read gesture
                GestureSample gesture = TouchPanel.ReadGesture();

                // is it Flick or Tap
                switch (gesture.GestureType)
                {
                    case GestureType.Flick:
                        // get drag delta amout
                        Single x = gesture.Delta.X, y = gesture.Delta.Y;

                        // dragged vertically or horizontally
                        if (Math.Abs(x) > Math.Abs(y))
                        {
                            // left or right
                            if (x < 0)
                            {
                                if (snake.direction == 3 && snake.snakeTiles.Count() >
1) Mode = GameMode.Complete;
                                else snake.direction = 1; // left
                            }
                            else
                            {
                                if (snake.direction == 1 && snake.snakeTiles.Count() >
1) Mode = GameMode.Complete;
                                else snake.direction = 3; // right
                            }
                        }
                        else
                        {
                            // up or down
                            if (y < 0)
                            {
                                if (snake.direction == 4 && snake.snakeTiles.Count() >
1) Mode = GameMode.Complete;
                                else snake.direction = 2; // up
                            }
                            else
                            {
                                if (snake.direction == 2 && snake.snakeTiles.Count() >
1) Mode = GameMode.Complete;
                                else snake.direction = 4; // down
                            }
                        }
                        break;
                    case GestureType.Tap:
                        // Game Mode from Menu to Running
                        if (Mode == GameMode.Menu)
                        {
                            Mode = GameMode.Running;
                        }
                        // Game Mode from Complete to Running again
                        else if (Mode == GameMode.Complete)
                        {
                            snake.Reset();
```


(C) Copyright Nokia 2014. All rights reserved.

```
                            score = 0f;
                            Mode = GameMode.Running;
                        }
                        break;
                }
            }
}
```

Drawing the Game

`Draw` method is called when the game should draw itself. In this example only Blue background color is used. Draw method checks here which is the Game Mode and draw the screen. "FreeSnake, Tap the Screen to Start" text will be drawn to the screen when game is loaded. Snake, food and score text will be drawn when the game is running. Game score and restart information is drawn when the game is over.

```
protected override void Draw(GameTime gameTime)
{
            GraphicsDevice.Clear(Color.CornflowerBlue);

            // Start drawing
            spriteBatch.Begin();

            if (Mode == GameMode.Menu)
            {
                string GameStart = "FreeSnake\nTap the Screen to Start";
                Vector2 stringSize = segoe20.MeasureString(GameStart);
                spriteBatch.DrawString(segoe20,
                    GameStart,
                    new Vector2((GraphicsDevice.Viewport.Width - stringSize.X) / 2,
(GraphicsDevice.Viewport.Height - stringSize.Y) / 2), Color.White);
            }
            else if (Mode == GameMode.Running)
            {
                // Draw the Snake
                snake.Draw(spriteBatch);
                // Draw the Food
                food.Draw(spriteBatch);
                // Draw the score
                spriteBatch.DrawString(segoe20, "Score:"+score, scorePosition,
Color.White);
            }
            else if (Mode == GameMode.Complete)
            {
                snake.Draw(spriteBatch);
                string GameOver = "Game Over\nScore : " + score +"\nTap the Screen to
Restart";
                Vector2 stringSize = segoe20.MeasureString(GameOver);
                spriteBatch.DrawString(segoe20,
                    GameOver,
                    new Vector2((GraphicsDevice.Viewport.Width - stringSize.X) / 2,
(GraphicsDevice.Viewport.Height - stringSize.Y) / 2), Color.White);
            }
            // Stop drawing
            spriteBatch.End();

            base.Draw(gameTime);
```


(C) Copyright Nokia 2014. All rights reserved.

```
}
```

Summary

This example shows the basic information to create Windows Phone Game with Gestures.

You can download source code here: File:PTMSnakeIt.zip.

(C) Copyright Nokia 2014. All rights reserved.