Farmer: Documentation

The solution is split into two main parts: the **Library** of classes which handle the actual game state and the **Graphics**.

Library

The game state is held within a **GameState** instance. The state keeps lists of **Farms** and **Coops**, which hold information about the current state of growing/chicken keeping. Also, the state keeps some temporary information needed for the running of the game, like the current **View**, current Farm/Coop, or held tool/product. Other than that, the state also holds information about player money, points, stamina and keeps instances of **Event** and **ChallengeHandler**, which take care of random events and challenges for the player. The state also handles buying and selling, and ending the day using **EndDay()**.

GameObject

All the objects in the game that are subject to the day cycle are subclasses of **GameObject**. The GameObjects are arranged in a hierarchy, where the GameObject they belong to takes care of calling **EndDay()** on them, starting at the GameState, which calls it on all its Coops and Farms. The system of GameObject is described below.

Farms

Each **Farm** holds instances of **Plot**. Each plot may or may not contain a **Plant** and forward farming interactions from tools to it. Plants keep track of their **GrowthState**, which determines how grown they are. Plants grow (or die) automatically at the end of the day based on watering, fertilization, presence of worms etc.

Coops

Each **Coop** contains a **ChickenFeeder** for its given maximum capacity of chicken, and then the actual instances of **Chicken** that are in the coop. It also takes care of feeding the chickens and emptying the feeder on DayEnd. Chickens lay eggs at the end of the day if they've been fed using **Feed()**. Eggs are laid into **EggSpots**, which are also GameObjects kept by the Coop.

Tools

The game has several kinds of tools, like **Pail** or **Hand**, which take care of specific farming interactions. They are all subclasses of **Tool**. They are used by calling the method **Use()** and passing the **IToolAcceptor** that the tool action should be done on. Each subclass implements the protected method **UseInternal()**, which defines the

actual tool action and is called from the wrapper Use() method. Things that support tool interaction (i.e. implement IToolAcceptor) are Plots, EggSpots and ChickenFeeders.

Buying and Selling

The buying and selling is handled by the GameState using the **ISellable** and **IBuyable** interfaces.

Each IBuyable has its **BuyPrice**, which is subtracted from the player money upon calling **Buy()** with it as a parameter. The GameState also doesn't allow negative money balance, so the sale won't be completed if the player doesn't have enough money. The state internally keeps count of how many of each IBuyable the player has, which can be accessed using the **GetOwnedAmount()** method. Classes that implement IBuyable are **Seed** classes and **Chicken**.

Selling is handled via the **SellHeld()** method, which sells the current **HeldProduct**, so to sell an ISellable, it must first be set as the HeldProduct. Each ISellable has its **SellPrice**, which is added to the player's money balance. Classes that implement ISellable are **Fruit** classes and **Eggs**.

Seeds and Fruits

Seed subclasses are ISellable and they take care of planting the corresponding **Plant** instances into **Plots**. This is done by calling the GameState's **PlantSeedToCurrent()** method, which plants the given Seed to all Plots in the current Farm.

Fruit subclasses are ISellable. They are collected by using the Hand tool on a **Plant** whose **GrowthState** is **Fruiting**.

Events

Events are things that have a certain chance of happening at the start of each day. Each possible event is represented by a **DayEvent** subclass. Each DayEvent has a determined chance of occurring and an effect. They are handled by the **DayEventHandler** class, which selects which DayEvents did occur, enacts their effects and returns a list of them using the **TryEvents()** method.

The possible events in the game are **RainEvent** (where everything is watered) and **WormEvent** (where more plants catch worms).

Challenges

Challenge subclasses represent individual types of challenges that the player may complete for points. They are handled by the **ChallengeHandler**, which check if they've been completed, returns points for the ones that were and replenished new challenges to replace them in the **CheckChallenges()** and **ReplenishChallenges()** methods.

Graphics

The graphics are handled by a central **FarmerGraphics** instance, which is tied to a Panel. This class keeps several **SceneHandler** subclasses, each of which handles the rendering of one scene of the game. Each SceneHandler draws its scene based on the provided **GameState**. SceneHandlers also forward mouse interactions to their control elements, which implement **IClickable**. Some (but not all) of these are subclasses of **GameButton**, which handles some basic tasks like checking the position of the mouse and highlighting the button on hover. The SceneHandlers also draw some top icons, like the **MoneyDisplay** or **StaminaDisplay**, on top of everything else. All of these must implement **IDrawable**, which means they are able to draw themselves based on a provided GameState. Some of these are also subclasses of **TextDisplay,** meaning they handle rendering some text based on the GameState (e.g. **MoneyDisplay** or **ProductTextDisplay** for handling the shop labels). The actual bitmaps to draw are loaded using different loader classes.

Buttons

The game buttons are subclasses of **GameButton**. The base class handles mouse positions, graphical rendering, hover behavior and enable/disable. Each button also has two (possibly empty) IClickable lists **ToEnable** and **ToDisable**; the button enables/disables them when clicked. Subclasses all implement the **Action** method, which determines the button's action on click (e.g. using a **Tool** on an **IToolAcceptor**). Some also override the **HoverAction** to add custom hover behavior.

Other Controls

Some of the controls are not GameButtons, but they only implement IClickable. Some of them are for custom behavior, like for the farm displays or the chicken feeder. Another notable control element is **MenuHandler**, which groups together GameButtons that act as one menu together.

CursorHandler

The state of the cursor is handled by the **CursorHandler**. It automatically displays the given GameState's **CurrentTool** or **HeldProduct** in the position of the mouse cursor (or nothing is both of them are null). It uses assets given to it via a **ToolIconLoader** instance and a **SellableLoader** instance.

Notes

The app only runs on Windows due to the use of the **Bitmap** class.