Список открытых методов, которые требуются от Arcomage.Core в Unity3d

1. Добавить игрока public void AddPlayer(TypePlayertp, string name)
2. Статистика игроков public Dictionary<Specifications, int> GetPlayerParams(SelectPlayer selectPlayer = SelectPlayer.None)
3. Получить карты игрока (противника) для отображения. Public List<Card> GetCard()
4. Имя победителя public string WhoWin()
5. Возможно ли использовать карту public bool IsCanUseCard(ICollection<CardParams> cardParams)
6. Информирование о случившемся событие, чтобы начало выполняться действие public bool GameNotification(Dictionary<string, object> information)

Публично доступные поля Arcomage.Core которые нужны в Unity

1. Состояние игры public CurrentAction status { get; private set; }

Список полей, которые используются в Arcomage.Core для внутренней работы:

1. Максимальное кол-во карт – public readonly int MaxCard;
2. Список игроков - private List<IPlayer> players { get; set; }
3. Текущий игрок private int currentPlayer { get; set; }
4. Параметры для выигрыша private readonly Dictionary<Specifications, int> WinParams;
5. Параметры проигрыша private readonly Dictionary<Specifications, int> LoseParams;
6. Стэк карт private Queue<Card> QCard = new Queue<Card>();

Список скрытых методов, которые используются в Arcomage.Core Arcomage.Core:

1. Начать новую игру. private void StartGame()
2. Генерация первичных параметров игроков private Dictionary<Specifications, int> GenerateDefault()
3. Использование карты private bool UseCard(int id)
4. Проверка, закончилась ли игра private bool isGameEnded()
5. Пропуск хода игроком private bool PassMove(int id)
6. Расчет ресурсов в конце хода игрока private CurrentAction EndMove()
7. Расчет хода компьютера private void MakeMoveAI()
8. Применение параметров карты к игроку private void ApplyCardParamsToPlayer(ICollection<CardParams> cardParams)
9. Применить параметры карты к противнику private void ApplyCardParamFromEnemy(CardParams item)
10. Применить прямой урон private void ApplyDirectDamage(CardParams item, int player)
11. Вычесть из статистики игрока значения указанные в карте private void MinusValue(Specifications spec, int value)
12. Прибавить/вычесть значения указанные в карте private void PlusValue(Specifications specifications, int value, int index)
13. Проверка, выиграл ли игрок private bool IsPlayerWin(Dictionary<Specifications, int> playerStatistic)
14. Проверка, проиграл ли игрок private bool IsPlayerLose(Dictionary<Specifications, int> playerStatistic)

Скажем так, системные параметры Arcomage.Core:

1. Логирование protected readonly ILog log;
2. Адрес веб-сервера private const string url = "http://kinglamer-001-site1.smarterasp.net/ArcoServer.svc?wsdl";
3. Интерфейс веб-сервиса private IArcoServer host;

Информация о том, какое действие необходимо выполнять сейчас:

Public enum CurrentAction

{

None, //Произошелпервыйзапуск

StartGame, //Начало игры

GetPlayerCard, //Получить карты для игрока

WaitHumanMove, //Ожидание хода игрока

HumanUseCard, //Игрок использовал карту

HumanCanPlayAgain, //Игрок может сходить еще раз

AnimateHumanMove, //Анимация использования карты игрока

UpdateStat, //Обновление статистики игроков

EndHumanMove, //Завершение хода игрока

AIMoveAnimation, //Анимация стола противника

AIUseCardAnimation, //Анимация использование хода противника

EndAIMove, //Завершение хода противника

EndGame//Завершение игры

}

Список методов, которые требуются в Unity3dдля обработки

* + 1. Прорисовка статистики игроков
    2. Прорисовка карт игрока
    3. Прорисовка рубашек карт противника
    4. Прорисовка использования карт игроком противника
    5. Запуск /Перезапуск игры
    6. Использование карты игроком
    7. Отображать кто выиграл и кнопки управления
    8. Возвращение неиспользованной карты игрока на место
    9. Обновление возможности использования карт

Какие события в Unity3dдолжны обрабатываться

1. Наведение на карту
2. Клик левой кнопкой мышки
3. Клик правой кнопкой мышки
4. Нажатие на Replay
5. Нажатие на Exit

Теперь рассмотрим более подробнее действия:

* + Получить карты для игрока

1. Обратиться к ядру и получить ровно столько карт, сколько необходимо для игрока
2. Отобразить их на поле игрока
3. Затемнить те карты, которые невозможно использовать, и они становится неактивны для выбора
4. Unityне должен знать, сколько ему необходимо отобразить карт. Это обязанность ядра
   * Ожидание хода игрока
5. Никаких изменений не должно происходить на столе. Пока не произойдет событие использование карты игрока
   * Анимация использования карты игрока
     1. Карта ожидает событий. Наведение мыши на карту – отображение увеличенной копии карты. Одиночное нажатие – карта используется.
     2. Нажатие правой клавишей – карта сбрасывается. Вызывается завершение хода игрока.
     3. Если карту невозможно использовать, прерывания события.
     4. Если карту возможно использовать, должно выполниться событие – обновление статистики игроков
     5. Потом обновление возможности использование карт
     6. Далее выполняется событие – Получить карты для игрока, но уже в данном случае новой карты
     7. Добавляется событие – Завершение хода игрока
   * Обновление статистики игроков
     1. Происходит повторный запрос, и перерисовка параметров игроков
   * Завершение хода игрока
     1. Проверяется может ли игрок сходить еще раз, если да, тогда вызывается событие ожидание хода игрока и текущее событие прерывается
     2. Карты становятся недоступными для выбора
     3. Выполняется Анимация стола противника
   * Анимация использование хода противника
     1. На стол вбрасывается та карту, которую противник использовал или сбросил
     2. Зависает на некоторое время, чтобы игрок успел ее разглядеть
     3. Разрушается
     4. Запускается событие – обновление статистики игроков
     5. Запускается событие – завершение хода противника
   * Завершение хода противника
     1. Карты становится доступными
     2. Выполняется – ожидание хода игрока
   * Завершение игры
     1. Все элементы на экране становятся неактивными
     2. Получаем с сервера имя победителя
     3. Отображаем его и 2 кнопки – Выход и Заново

Алгоритм пропуска хода игроком

В данном алгоритме мы начнем с состояния **WaitHumanMove** - ожидание хода игрока

1. если пропустить ход, тогда сбрасываем карту из руки игрока (List.RemoveAt(index)) и меняем на состояние **PassStroke**.
2. Ожидание пока игра сбросит карту и вызовет метод gm.GameNotification
3. Текущий статус у нас **PassStroke** - он получает подтверждение, что карта скинута, выполняет метод UpdateStat для текущего игрока и меняет статус на UpdateStat
4. В игре происходит перерисовка статистика игрока и об этом уведомляется ядро.
5. Текущий статус UpdateStat. Ядро делает статус EndHumanMove.
6. В игре происходит блокировка контролов игрока, до тех пор, пока не наступит статус WaitHumanMove.
7. Текущий статус EndHumanMove. В зависимости от противника (это нужно будет потом учесть) идет переход хода. В нашей ситуации, ход передается компьютеру. Выполняется метод AIMove. Который вовзращает так же уведомление о том, что сделал комп, сбросил или использовал карту. Меняет на статус AIUseCardAnimation. Вызывается рекурсивно метод gm.GameNotification
8. В игре при проверке получает информацию о том, что текущий статус AIUseCardAnimation и нужно отрисовать карту. Так же есть нюанс, карта используется или сбрасывается (это разные анимации). Следовательно, игра обращается к ядру за получением информации о карте использованной компьютером и как ее использовал (сейчас этого нет). Отрисовывет ее соответствующим образом и уведомляет ядро о том, что прорисовка закончена.
9. Текущий статус AIUseCardAnimation - ядро получает подтверждение, что карта отображена для игрока, выполняет метод UpdateStat для текущего игрока и меняет статус на UpdateStat (вот второй нюанс, нужны разные статусы для обновления, т.к. из этого статуса потом нужно перейти в нужны метод)
10. В игре происходит перерисовка статистика игрока и об этом уведомляется ядро.
11. Текущий статус UpdateStat. Ядро делает статус EndAIMove.
12. В игре происходит снятие блокировки контролов игрока, идет отсылка в ядро, что все объекты разблокированый
13. Текущий статус EndAIMove. Получаем уведомление, что разблокирован игрок, меняем статус на GetPlayerCard



Автомат состояний

Состояние 1 (Стартовое)

В ядре. Получение 5 карт игрока с сервера.

В игре действий нет.

Состояние 2

В ядре. Получение 5 карт ПК с сервера.

В игре действий нет.

Состояние 3

В ядре. Передача 5 карт в Игру.

В игре – прорисовка 5 карт игрока.

Состояние 4

В ядре. Выбор игрок или ПК делает первый ход.

В игре действий нет.

Состояние 5

В ядре. Передача в ИИ данных о картах ПК.

В игре. ИИ просчитывает свой ход и выбирает карту для игры или для сброса (Пас). Информация Игра или Пас отправляется в ядро. Информация(id) по карте отправляется в ядро.

Состояние 6

В ядре. Состояние появляется если в С5 ИИ выбрал Игра. Действий нет.

В игре. Анимация хода ПК. Появляется сыгранная карта, и фиксируется на поле в спец месте. Затем (возможно) исчезает – растворяется.

Состояние 7

В ядре. Состояние появляется если в С5 ИИ выбрал Пас. Действий нет.

В игре. Анимация Пас для ПК. Выводится надпись что ПК сбросил карту.

Состояние 8

В ядре – обновление статистики.

В игре – обновление статистики

Состояние 9

В ядре – удаляется сыгранная карта. Если на карте есть флаг Повторный ход, то получение 1 карты с сервера И переход в состояние С4. Если флага нет, то переход в состояние С10

В игре – действий нет.

Состояние 10

В ядре – нет действий.

В игре – ожидание действий игрока. Неиспользуемые карты затемнены. При наведении отображается увеличенная копия карты. Ход картой – левая кнопка мыши. Сброс карты – правая кнопка мыши. Информация Игра или Пас отправляется в ядро. Информация(id) по карте отправляется в ядро.

Состояние 11

В ядре – состояние появляется если игрок в С10 нажал на карте левую кнопку мыши(Игра).

В игре анимация хода. Карта перемещается в специальное место и фиксируется там. Затем (возможно) исчезает – растворяется.

Состояние 12

В ядре – состояние появляется если игрок в С10 нажал на карте правую кнопку мыши (Пас).

В игре анимация Пас. Карта исчезает – растворяется.

Состояние 13

В ядре – обновление статистики.

В игре – обновление статистики

Состояние 14

В ядре – удаляется сыгранная карта. Если на карте есть флаг Повторный ход информация об этом сохраняется в спец флаг. С сервера получается 1 новая карта для игрока.

В игре действий нет.

Состояние 15

В ядре – если в С14 был выставлен флаг повторного хода то сброс этого флага и переход в состояние С10, если нет то переход в С5. Данные по карте передаются в игру(возможные параметры: id, название, описание, цена, тип и пр.).

В игре – прорисовка карты на месте уничтоженной в С11 или С12.