

Specyfikacja gry komputerowej

Arcade Rally

Michał Kafka



Wydział Automatyki Elektroniki i Informatyki

Politechnika Śląska

3 kwietnia 2020

1 Opis zasad gry

Arcade Rally to prosta gra z gatunku gier wyścigowych i symulacji rajdów samochodowych. Zadaniem gracza jest pokonać odcinek specjalny w jak najkrótszym czasie. Standardowy warunek zwycięstwa znany z innych gier w tym przypadku nie istnieje, gdyż punktem odniesienia dla gracza jest czas, który ewentualnie można poprawić w kolejnych próbach.

Gra umożliwia poruszanie się jednym modelem samochodu po pojedynczym odcinku specjalnym, składającym się z czterech sektorów. Czas jest liczony nie tylko dla całego odcinka, ale również dla każdego punktu kontrolnego.



Rysunek 1: Zrzut ekranu z gry *Arcade Rally*.

2 Specyfikacja zewnętrzna

Po uruchomieniu pliku wykonywalnego gry rozpoczyna się odliczanie do startu. Po pięciu sekundach stoper rusza, a gracz może kontrolować samochód. Sterowanie polega na wykorzystaniu klawiszy W, S, A i D lub strzałek w przód, w tył, w lewo i w prawo, które umożliwiają przyspieszanie, hamowanie oraz manewrowanie pojazdem. Klawisz R umożliwia włączenie i wyłączenie biegu wstecznego, przy czym należy pamiętać, że jest możliwe jedynie gdy prędkość samochodu jest bardzo niska.

Poza samym pojazdem gracz ma również możliwość zarządzania rozgrywką poprzez klawisze G i B. Pierwszy z nich umożliwia zresetowanie rozgrywki, zaś drugi pozwala wrócić na tor w razie utknięcia poza trasą. Wyjście z gry jest możliwe za pomocą kombinacji klawiszy Alt i F4.



Rysunek 2: Zrzut ekranu z gry *Arcade Rally*. Samochód zbliża się do kolejnego punktu kontrolnego.

3 Specyfikacja wewnętrzna

Wśród kluczowych elementów programu należy przede wszystkim wyróżnić dwa skrypty odpowiadające za poruszanie się samochodu i jego zawieszenia, czyli: *CarMovement.cs* i *WheelMovement.cs*.

Pierwszy skrypt definiuje fizykę jazdy, a zatem dotyczy on takich elementów jak zmiana biegów, pomiar prędkości i obrotów, mechanika hamulców i gazu, rozłożenia napędu itd. Poniżej znajduje się listing 1, który prezentuje fragment kodu odpowiedzialny za zmianę przełożeń, warto zauważyć, że zależy ona nie tylko od obrotów silnika, ale również od prędkości pojazdu, dzięki czemu bieg nie jest zmieniany na wyższy, gdy koła buksują przy niskiej prędkości. Ponadto, wprowadzono zmienną *gearShiftTimer*, która pełni funkcję sprzęgła, co pozwala na kontrolowanie mocy przekazywanej na koła podczas zmiany biegu i zapobiega gwałtownym skokom mocy i przełożeń.

```

1 private void UpdateGear() {
2
3     if (gearShiftTimer >= 0){
4         gearShiftTimer -= Time.deltaTime;
5         return;
6     }
7
8     if (speed < reverseShiftSpeed && Input.GetKeyDown(KeyCode.R
9 )) {
10         reverse = reverse ? false : true;
11         gear = 1;
12         gearShiftTimer = gearShiftTime;
13     }
14
15     // Do not shift gears if reverse is on
16     if (reverse == true) {

```

```

16         return;
17     }
18
19     if (rpm > shiftPoint * maxRpm + shiftMargin && gear <
20         gearRatios.Length && speed > gearUpshiftSpeeds[gear - 1]){
21         gear += 1;
22         gearShiftTimer = gearShiftTime;
23     }
24     else if (rpm < shiftPoint * maxRpm - shiftMargin && gear >
25         1 && speed < gearDownshiftSpeeds[gear - 1]){
26         gear -= 1;
27         gearShiftTimer = gearShiftTime;
28     }

```

Listing 1: Metoda odpowiedzialna za kontrolowanie skrzyni biegów samochodu.

Skrypt *WheelMovement.cs*, jak już wspomniano, kontroluje pracę zawieszenia. Dzięki niemu, koła utrzymują kontakt z podłożem i reagują na prędkość obrotową pólci, a także ruchy wirtualnej kierownicy. Listing 2 przedstawia fragment kodu odpowiedzialny za zarządzanie położeniem koła względem gruntu. Wykorzystano w nim właściwości elementów silnika fizycznego Unity *WheelCollider* oraz *Raycast*.

```

1 private void HandleWheelPosition(GameObject mesh, Vector3
2     localBasePos, WheelCollider wheelCollider) {
3
4     float radius = wheelCollider.radius;
5
6     RaycastHit hit;
7     Physics.Raycast(wheelCollider.transform.position, -Vector3.
8         up, out hit);
9
10    float distance = Vector3.Distance(wheelCollider.transform.
11        position, hit.point);
12    float offset = radius - distance;
13
14    if (wheelCollider.isGrounded) {
15        mesh.transform.localPosition = localBasePos + new
16        Vector3(0, offset + wheelCollider.suspensionDistance / 2, 0)
17        ;
18    }
19    else {
20        mesh.transform.localPosition = localBasePos + new
21        Vector3(0, -wheelCollider.suspensionDistance / 2, 0);
22    }
23 }

```

Listing 2: Kod kontrolujący pracę zawieszenia wirtualnego samochodu.

Pozostałe skrypty to *FollowingCamera.cs*, *Checkpoint.cs*, *RaceManager.cs* i *WaypointManager.cs*, które odpowiadają za pracę kamery oraz elementy związane z rozgrywką wymienione w sekcji 2 niniejszego dokumentu.

4 Testy

W ramach tworzenia programu nie przeprowadzono żadnych specyficznych testów jednostkowych czy też testów integracyjnych. Gra *Arcade Rally* jest w fazie wczesnego rozwoju i testowanie wdrażanych funkcjonalności głównie opiera się na subiektywnych odczuciach osób testujących.

5 Podsumowanie

Zgodnie z wymaganiami gra *Arcade Rally* implementuje elementy silnika fizycznego Unity m.in. kolizje, wyzwalacze i wirtualne koła, a także uwzględnia komponenty terenu. Warto zauważyć, że świat gry nie tylko składa się z trawy i drzew posadzonych na powierzchni obiektu *Terrain*, ale również z toru, który został wygenerowany w programie Blender 3D.