TechCity Kodų Karai, II Epizodas – API Specifikacija

Dokumento tikslas

Šis dokumentas yra skirtas TechCity Kodų Karai, II Epizodas dalyviams ir negali būti paviešintas kol neprasidėjo Kodų Karų turnyras. Pagrindinis dokumento tikslas – dokumentuoti kliento/serverio sąsają bei GameHandler adapterį, kurio pagalba sukurtos programos keisis duomenimis su centriniu žaidimų serveriu.

Kliento/serverio sąsaja ir Game Handler

Anksčiau vykusiose varžybose komandos turėjo įgyvendinti visą reikalingą komunikaciją su serveriu tiesiogiai per oficialų API, tačiau daugeliui komandų tai kėlė labai daug rūpesčių ir reikalavo neproporcingai daug laiko atiderinti. Atsižvelgiant į pateiktas dalyvių pastabas ir komentarus, turnyro organizatoriai nusprendė pateikti oficialų adapterį – Game Handler.

Game Handler (trumpai – GH) yra Node.js skriptas, skirtas integracijai tarp komandų pateikiamų sprendimų ir žaidimo serverio. Šis skriptas supaprastina visą komunikaciją su serveriu, o komandoms belieka įgyvendinti tik vieną funkciją – žaidimo ėjimo pateikimą.

Techniniai reikalavimai Game Handler naudojimui

GH sudaro vienintelis failas – *gh.js*. Jam paleisti reikia turėti instaliuotą Node.js. Standartinėje instaliacijoje nesantys paketai yra pateikiami kartu su sprendimu *node_modules* direktorijoje, taigi teoriškai papildomai nereikės nieko instaliuoti. Pateikta konfigūracija patikrinta su **Node.js LTS versija iš v6 serijos**.

Node.js ir JavaScript sprendimas buvo pasirinktas tam, kad būtų maksimaliai išplėstos dalyvių galimybės naudoti jų pačių pasirinktą programavimo aplinką.

Game Handler darbo režimai

Kai tik *gh.js* yra paleidžiamas, jis iš karto jungiasi prie žaidimo serverio ir tęsia darbą tol, kol bus nutrauktas rankiniu būdu arba įvykus nepataisomai klaidai. GH paleidimo metu yra nurodomas konfigūracijos failas, kuriame yra visa informacija reikalinga prisijungimui prie žaidimo serverio. Taip pat konfigūracijoje yra nurodoma, kaip susisiekti su komandos parašytu sprendimu.

Komandos sprendimas gali būti integruotas dviem būdais: ***exec*** ir ***json***.

- ***exec*** režime žaidimo informacija apsikeičiama per failus. Kiekvieno ėjimo metu aktuali žaidimo būsena yra įrašoma į failą *sessions* direktorijoje, tuomet yra paleidžiama sprendimo programa. Jūsų pateikta programa perskaito šį failą ir ėjimo sprendimą įrašo į kitą nurodytą failą. Kai paleista programa baigia darbą, GH perskaito sugeneruotą sprendimo failą ir perduoda jo turinį į serverį. ***exec*** režimas veikia lėtai, tačiau jį gali būti lengviau ir greičiau įgyvendinti. Be to, veikimo metu galima stebėti visą duomenų apsikeitimo istoriją, kas gali padėti pradiniuose programos derinimo etapuose.
- ***json*** sprendimas kiekvieno ėjimo metu aktualią žaidimo informaciją persiunčia kaip HTTP POST turinį į konfigūracijoje nurodytą URL, o gautą užklausos atsakymą perduoda atgal į žaidimo serverį. Šis režimas veikia žymiai greičiau ir leidžia jūsų sprendimui paprasčiau išlaikyti duomenis tarp ėjimų ar net žaidimų.

Galutiniam sprendimui yra **rekomenduojamas *json* darbo režimas**.

Konfigūracija ir paleidimas

GH yra leidžiamas iš komandinės eilutės:

```
# node gh.js profile.json ClientName
```

Profile.json – tai konfigūracijos failo vardas, o ClientName yra nebūtinas parametras, nurodantis, kokiu vardu matysite šį žaidėją. Visi klientai privalo turėti unikalius vardus, kitaip jie trukdys vienas kitam dirbdami su žaidimo serveriu. Nesistenkite generuoti ypatingai unikalių vardų – sugalvokite savo klientui vardą ir naudokite jį pastoviai. Būkite išradingi, bet nepamirškite, kad kliento vardus galiausiai matys visi varžybų dalyviai.

Jei kliento vardas nenurodytas paleidimo metu, jis bus imamas iš konfigūracijos failo.

Konfigūracijos failo struktūra

GH konfigūracijos failas turi tokią struktūrą:

```
{
  "serverUri": "http://server-name/ClientService.svc",
  "teamName": "Mano komanda",
  "teamPassword": "komandos slaptažodis",
  "clientName": "Petras",
…
```

}

Parametras	**Aprašymas**
serverUri	Žaidimo serverio adresas (suteikiamas organizatorių)
teamName	Oficialus komandos pavadinimas (suteikiamas organizatorių)
teamPassword	Komandai suteiktas slaptas kodas (suteikiamas organizatorių)
clientName	Paleidžiamos programos žaidėjo vardas, kuris bus matomas žaidėjų sąraše.
Režimo konfigūracija (galima viena iš dviejų arba ***exec***, arba ***json***)	
Exec režimas	
clientType	exec
execName	path\\to\\your_program.exe Nurodykite kelią iki paleidžiamos programos. Naudokite Jūsų OS tinkamą formatą (šis pavyzdys yra skirtas Windows sistemai – atkreipkite dėmesį dvigubus kelio skirtukus) Žaidimo duomenų ir atsakymo failai bus laikomis *sessions* direktorijoje, o jų vardai bus perduodami paleidžiamai programai kaip parametrai komandinėje eilutėje. Failų direktorija gali būti pakeista konfigūracijos parametru *sessionDir*.
Json režimas	
clientType	json
jsonUri	http://localhost/path/to/your_service

Duomenų Formatas

Žemėlapis

Žemėlapis yra stačiakampio formos ir kiekviena pozicija jame žymima tokiais simboliais:

- Tarpas – tuščias langelis, žiurkės gali judėti ir būti šiuose langeliuose nevaržomos.
- Taškas (.) – langelis su sausainiu.
- Grotelės (#) – siena. Paprastai šiuose langeliuose judėti draudžiama, bet jei kuri žiurkė čia pajudėtų, siena nugriaunama, o žiurkė sunaikinama.

Skaitmenys (0..7) – žymi tam tikro žaidėjo žiurkių startines pozicijas. Paprastai naudojami tik 0 ir 1 dviejų žaidėjų atveju. Šios žymės yra tik pradiniame žemėlapyje, kuris pakraunamas prieš žaidimo pradžią – žaidimo metu šios pozicijos tampa tuščios

Ėjimo informacija

Input.json

Kiekvieno ėjimo metu žaidėjų programa gauna tokio formato informaciją:

```
class PlayerReq
{
    string GameUid;
    int Turn;
    EnMapData Map;
    EnPlayerState[] Players;
    int YourIndex; // Nurodo jūsų žaidėjo indeksą Players masyve
    int LastTurn;  // Kai Turn==LastTurn, vyksta paskutinis žaidimo ėjimas
}

class EnMapData
{
    int Width;
    int Height;
    string[] Rows; // Visos žemėlapio eilutės
}

class EnPlayerState
{
    string Condition;
    string Comment;
    int Score;
    EnRatPosition[] RatPositions; // Žaidėjo (gyvų) žiurkių pozicijos
}

public class EnRatPosition
```

```
{
    int RatId; // Identifikuoja žiurkę ėjimo metu, nesikeičia vieno žaidimo ribose
    EnPoint Position;
}

public class EnPoint
{
    int Row;
    int Col;
}
```

Output.json

Kiekvieno ėjimo metu žaidėjo programa turi perduoti tokio formato informaciją:

```
class PlayerResp
{
    EnPlayerMove[] Moves;
}

class EnPlayerMove
{
    string Action;
    int RatId;
    EnPoint Position;
}
```

Kiekvienai gyvai žiurkei privaloma perduoti šią informaciją apie jos vykdomą ėjimą:

- Judėjimas:
    - Action – „Move“
    - Position – nauja žiurkės pozicija (gali sutapti su dabartine, kad žiurkė liktų vietoje).
- Sausainio valgymas:
    - Action – „Eat“
- Susisprogdinimas:
    - Action – „Explode“

RatId visuomet nurodomas žiurkės identifikatorius. Position lauką privaloma nurodyti tik judėjimo metu.