**情報システム基礎演習**

チーム開発のための版管理システム

締切日　6/14

提出日　6/13

3班　B15077 西田　章馬　e1b15077

## Github Flowにもとづくじゃんけんゲーム開発

今回の演習では，2つのじゃんけんゲームを開発した．そのうち，自分が主開発者となったものについて，共同開発者の分まで含めて振り返りを実施し，以下に記述すること．

## 開発者情報

|  |  |
| --- | --- |
| 主開発者 | 共同開発者 |
| 氏名(自分)：西田　章馬  学番(自分)：B15077  Github(自分):e1b15077 | 氏名：榎本　真也  学番：B15020  Github:enoda1002 |

## リポジトリURL

主開発者のGitリポジトリURLを下記URLに続けて記入せよ．

https://github.com/e1b15077/Janken\_B15077

## 開発状況

主開発者として開発したじゃんけんゲームの動作画面をキャプチャし，以下に貼り付けよ．また，正常系シナリオ(S0~D8)/異常系シナリオ(E1~E12)のどこまで完了できたかを記述せよ．

[じゃんけんゲーム動作画面]

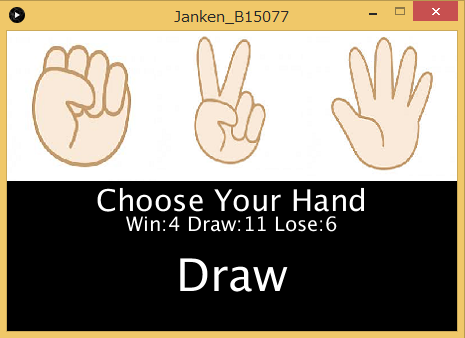


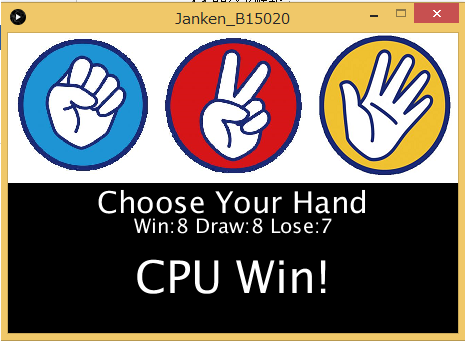
図1　正常系シナリオの動作画面

図2　異常系シナリオの動作画面

[開発進捗状況]

正常系シナリオ　S0~D8

異常系シナリオ　E1~E9

## ブランチ名の確認

主開発者として開発したじゃんけんゲームリポジトリにおいて，masterブランチ以外に作成されたすべてのブランチ名を以下に記述せよ．さらに，すべてのブランチ名が適切であるか（本来はそのブランチで何をするかが第3者に伝わる名前であるかを評価することが目的だが，ここでは演習中で指示されたものと一致しているかどうかを記述すること）．なお，ブランチ名は 2.で記述したGitリポジトリURL末尾に/branches/all をつけることでも表示できる．

add\_total\_score

add\_game\_result\_display

add\_janken\_display

[ブランチ名に関する考察]

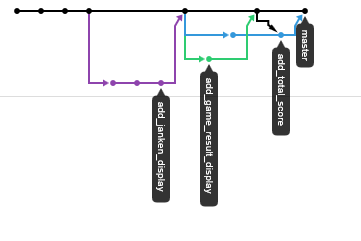
ブランチを活用すると，複数人で同じ開発の異なった作業を並行して進めることができる．よって，ブランチ名には作業内容がわかる名前をつけることで開発を円滑に進めることができると考える．

## コミットグラフ

主開発者として開発したじゃんけんゲームリポジトリにおいて， Githubサイトで表示されるコミットグラフをキャプチャし，以下に貼り付けよ．コミットグラフは，2. で記述したGitリポジトリURL末尾に"/network"をつけることで表示できる．  
その後，下記1~3それぞれについて対応できているかを記述せよ．

1. すべてのブランチがmasterブランチから作成されているか  
2. すべてのブランチがMergeされているか  
3. 初期コミット以外でmasterブランチに直接コミットされていないか（Pull RequestによるMergeならOK）

[コミットグラフ]



1~3すべての条件を満たしている．

[ブランチ及びコミットに関する考察]

Githubでは，masterからブランチを切ることで，異なった作業を並行して行うことができ，作業が終了するとmasterにPull Requestを投げ，第三者のレビューを複数受けてからmasterへMergeする．ブランチ間でコミットによる影響はなく，「保存」という考えでコミットを定期的に行うのが好ましい．それにより，作業内容の損失などを防ぐことができるため，定期的にコミットを行うのは開発を円滑に進める上では必要だと考える．

## Pull Request

主開発者として開発したじゃんけんゲームリポジトリにおいて，Pull Requestの情報を下記に記述し，1,2に対応できているか確認し，記述せよ．Pull Requestは2. で記述したGitリポジトリURL末尾に"/pulls"をつけることで表示できる．Mergeが終了したPull RequestはClosedというリンクで，MergeされていないPull RequestはOpenというリンクをクリックすることで表示できる．

1.すべてのPRがMergeされているか  
2.PR登録者とレビュアーが別の開発者になっているか

|  |  |  |  |
| --- | --- | --- | --- |
| PR URL(#) | タイトル | PR作成・登録者 | レビュアー |
| #1 | じゃんけん機能追加 | e1b15077 | enoda1002 |
| #2 | 結果表示 | enoda1002 | e1b15077 |
| #3 | スコア機能の追加 | e1b15077 | enoda1002 |
|  |  |  |  |

ここで，PR URLにはGithubで割り当てられたPull Requestの#を記述すれば良い(#1,#2等)．PR作成・登録者及びレビュアーは担当した開発者のGithub アカウントを記述すること．

1と2に対応している．

[Pull Requestに関する考察]

Pull Requestは自分で変更したブランチの内容を主開発者のmasterブランチに取り込んでもらえるように依頼するのに使用し，Pull Requestを投げると投げられた相手は変更したコードなどを参照することができ，第三者の目でレビューを行うことができる．間違っているコードや効率の悪いコードをmasterに取り込まないためにも必要な作業で，簡単なコメントも書き込むことができるので，開発場所が離れていてもコミュニケーションをとることができる．

## 発生したトラブル及び得られた知見，開発における改善点

今回のじゃんけんゲームでは主に目立った改善点は見当たらなかったが，コミットやPull Requestを行うときに，タイトルはしっかりと書けているがコードの変更部分や機能追加を明記するとよりわかりやすく効率よく開発ができると考える．また，仕方がない問題ではあると思うが，ブランチ作成やコードを書き換えたりした後のプッシュ作業や，sync作業の時間が減少すれば開発にかかる時間も削減できると考える．

## 5. Github Flowにもとづくどうぶつしょうぎ開発

どうぶつしょうぎの開発について，自身の行動を下記にしたがって振り返ること．なお，本シートはそのまま小レポートの一部となる．

## 開発者情報

主開発者及び共同開発者の情報を下記に記述すること．また，自分の学生番号，氏名，Githubアカウントについてはアンダーライン(“\_\_\_”)を引いて分かるようにしておくこと．

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 主開発者 | 共同開発者1 | 共同開発者2 | 共同開発者3 |
| 学生番号 | B15020 | B15077 | B15038 | B15094 |
| 氏名 | 榎本　真也 | 西田　章馬 | 小池　志門 | 松浪　貴之 |
| Github | enoda1002 | e1b15077 | e1b15038 | e1b15094 |

## 5.2リポジトリURL

どうぶつしょうぎのGitリポジトリURLを下記URLに続けて記入せよ．

https://github.com/enoda1002/Doubutsushougi

## 5.3 開発状況

どうぶつしょうぎの動作画面をキャプチャし，以下に貼り付けよ．また，全9Stepのどこまでできたかを記述せよ． Step内の特定のTaskまでの場合，どのTaskまでできたかもあわせて記述すること．

[どうぶつしょうぎ動作画面]

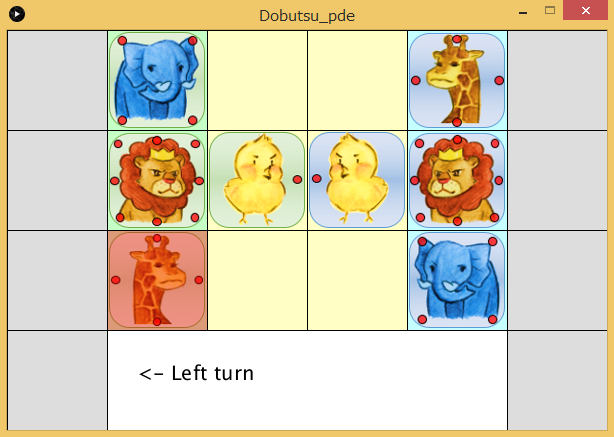


図3　どうぶつしょうぎの動作画面

[開発進捗状況]

step3まで開発完了

## 開発担当Task数とレビュー担当Task数の確認

自分が担当したTaskを開発，レビューそれぞれについて記述せよ．さらに，最後にそれぞれの担当Task数を記述せよ．なお，開発・レビューのどちらか一つでも担当Task数が5に満たない場合は，開発のやり直しが発生することがあるので，チームメンバ及び教員に相談すること．

|  |  |
| --- | --- |
| 自分が開発を担当したTask | 自分がレビューを担当したTask |
| Task1-1  Task1-3  Task2-2-b  Task2-2-f  Task3-2  開発担当Task数：5 | Task1-2-c  Task2-2-a  Task2-2-e  Task2-4  Task3-1  レビュー担当Task数：5 |

## ブランチ名の確認

自分の作成したブランチ名とそのブランチでの開発内容を以下に記述せよ．なお，ブランチ名は 2.で記述したGitリポジトリURL末尾に/branches/all をつけることでも表示できる．

[自分の作成したブランチ名及びブランチでの開発内容一覧]

Task1-1 AbstractAreaクラスの作成

Task1-3 Boardクラスの作成

Task2-2-b Zouクラスの作成

Task2-2-f コマ画像データの登録

Task3-2 GameStatusクラスの作成

次に，自分が作成したブランチ名について，第3者に開発内容が伝わる名前になっているかを考察し，記述せよ．わかりにくい名前になっていると思ったものについては，良いブランチ名とあわせて記述すると良い．

[ブランチ名に関する考察]

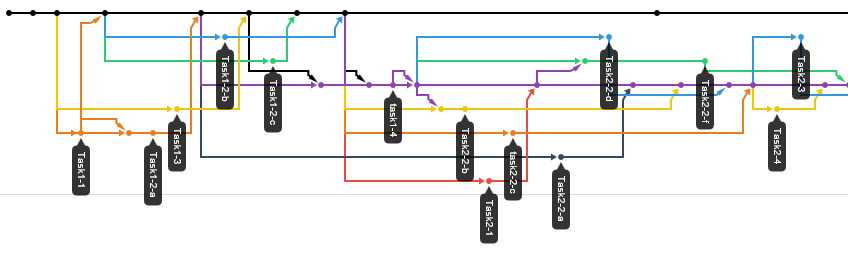
今回は与えられた課題がTaskごとに分かれていたので，このような名前にするとチームで決定し開発を進めたが，チーム以外の第三者から見ると，どのような開発を行っているのかわからない．よって，Taskの番号をブランチ名にするのではなく，実装した機能などの名前をブランチ名にすると，第三者にもわかりやすい開発ができると考える．

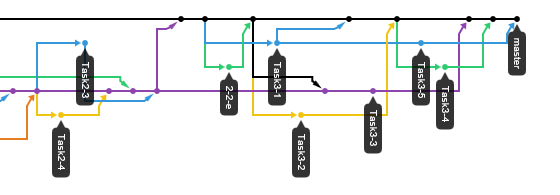
## コミットグラフ

どうぶつしょうぎリポジトリにおいて， Githubサイトで表示されるコミットグラフをキャプチャし，以下に貼り付けよ．その後，下記1~3それぞれについて対応できているかを記述せよ．コミットグラフは，2. で記述したGitリポジトリURL末尾に"/network"をつけることで表示できる．これについては，自分が作ったブランチであるかどうかを考慮せず，すべてのコミットグラフを貼り付けて良い．

1.すべてのブランチがmasterブランチから作成されているか  
2.すべてのブランチがMergeされているか  
3.初期コミット以外でmasterブランチに直接コミットされていないか（Pull RequestによるMergeならOK）

[コミットグラフ]





1~3の条件を満たしていないブランチが存在する．

Task2-2-d，Task2-3はMergeがされてなく，masterブランチから作成されていないブランチが多い．なぜこのようなことになったかというと，masterに直接コミットするミスが起こってから班内で混乱が起こり，他のブランチをmasterブランチだと勘違いしてそのままコミットMergeしたことが原因だと考える．

[ブランチ及びコミットに関する考察]

今回の開発は前回とは違い，開発人数が増えたためブランチの数がとても多くなった．チーム内でブランチの名前の付け方を決定していたので開発者視点から見れば見やすくまとまっていると考える．メンバーの一人がmasterブランチでコードを書き，コミットしてしまったのが今回の開発ではよく目立ち，チーム開発が混乱した原因である．masterブランチにあるコードは常に実行可能で，しっかりと動くものでなければいけないが，今回のようにmasterブランチでコードの変更を行うと，システムが急に動かなくなったりバグの発見改変に時間がかかるので，開発にかかる時間が増えてしまうので気を付けなければいけない点である．

## Pull Request

今回自分が作成・登録したPull Request及び自分がレビューを行ったPull Requestについて，下記に記述し，以下の1~3を守れているか考察すること．特に，3のPull Request作成・登録数が2に満たない場合は，開発のやり直しが発生する場合があるため，チームメンバ及び教員に連絡すること．Pull Requestは2. で記述したGitリポジトリURL末尾に"/pulls"をつけることで表示できる．Mergeが終了したPull RequestはClosedというリンクで，MergeされていないPull RequestはOpenというリンクをクリックすることで表示できる．

1.すべてのPRがMergeされているか  
2.PR登録者とレビュアーが別の開発者になっているか  
3.Pull Request作成・登録数が2以上である

|  |  |  |  |
| --- | --- | --- | --- |
| PR URL(#) | タイトル | PR作成・登録者 | レビュアー |
| #1 | AbstractAreaクラスの作成 | e1b15077 | enoda1002 |
| #3 | Boardクラスの作成 | e1b15077 | e1b15038 |
| #5 | 1-2-c | e1b15038 | e1b15077 |
| #7 | Task2-2-b | e1b15077 | enoda1002 |
| #10 | 2-2-a | e1b15038 | e1b15077 |
| #12 | Task2 2 f | e1b15077 | enoda1002 |
| #13 | Doubutsuの更新 | e1b15094 | e1b15077 |
| #15 | 2-2-e | e1b15038 | e1b15077 |
| #16 | AbstractKomaの更新 | e1b15038 | e1b15077 |
| #17 | Task3-2 | e1b15077 | enoda1002 |

ここで，PR URLにはGithubで割り当てられたPull Requestの#で良い(#1,#2等)．PR作成・登録者及びレビュアーはGithub アカウントを記述すること．

1~3のすべての項目を満たしている．

[Pull Requestに関する考察]

今回の開発では4人が行うレビュー・Pull Requestを均等にわけることができ，既定の数行っているのでノルマは達成できた．1回目授業で開発していたリポジトリは，途中でmasterブランチがデプロイできなくなったため，一から違うリポジトリを作り直し，幸いその時の開発はレビューやPull Requestの数がチーム内で均等ではなかったので，リポジトリを作り直した正解であった．

## 発生したトラブル及び得られた知見，開発における改善点

Githubでは，masterブランチが常にデプロイ可能な状態にしておかないといけないので，ブランチを切りそのブランチ内で様々な開発を行うのが好ましい．またブランチ名は，第三者が見て何の開発を行っているのかがわかる名前を付けたほうが，効率よく開発を行うことができる．レビューするのは一人ではなく，多数のほうがバグの発見や効率の悪いコードの発見を見逃すのを防止できるため，チーム全体でレビューを行うのが好ましい．

8. おわりに

今回の開発を通してわかったことは，Githubは使い勝手が良いが，班員全員が使い方をしっかりと理解しておかないと今回のような，masterブランチに直接コミットするだとかmasterブランチから他のブランチが作成されていないなどの致命的なミスが起こりやすい．今回は小さな開発なので良かったものの，企業の開発でこのようなことが起これば大変な事態だと考える．開発者間のコミュニケーションを円滑にする開発手法だが，そのコミュニケーションや連携力が足りなかったため，一応プログラムは動くが今回の開発は失敗と言ってよいと考える．

以下を修正して再提出すること（6/1）

4.6

「Pull Requestの情報を下記に記述し，1,2に対応できているか確認し，記述せよ」に対応すること

5.6

条件1,2が満たせていないブランチが存在する．具体的なブランチ名も含めてなぜそうなったかと合わせて記述すること．

5.7

「以下の1~3を守れているか考察すること」が言及されていない．記述すること．

小レポートの作成要領について以下のように書かれているが，8がない．

「上述した大レポートの「4.Github Flowにもとづくじゃんけんゲーム開発」「5.Github Flowにもとづくどうぶつしょうぎ開発」と8.おわりにを参考にレポートを作成すること．」