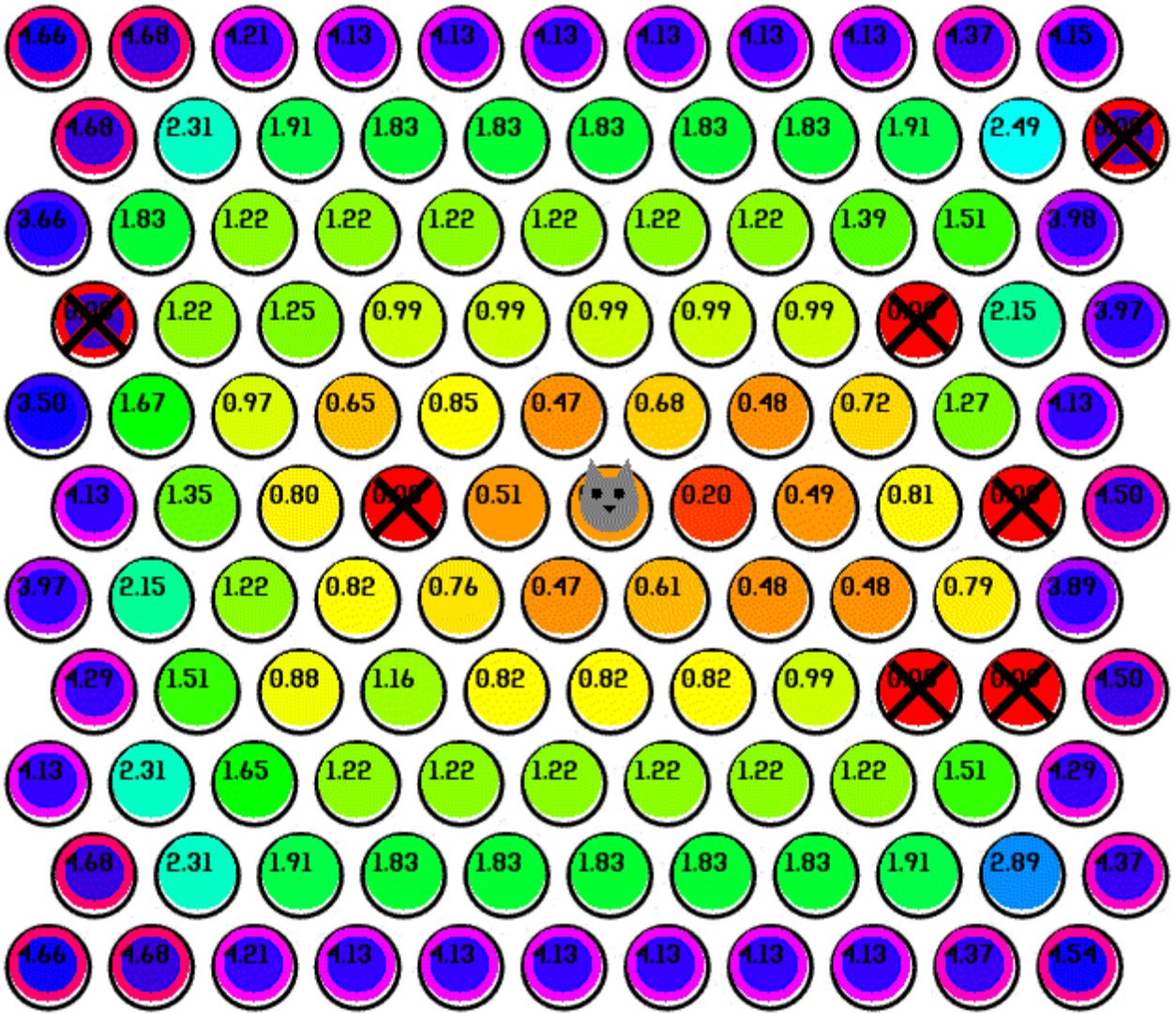
Resolução de Problemas - *Cat* e *Catcher*

Gato

Estrutura, comportamento e código principal

```
root
|   cat.py [arquivo que controla os movimentos do gato e a escolha do próximo passo]
|   scores_calculator_cat.py [arquivo que atribui scores a pontos do tabuleiro]
|
+---cats [arquivos para calcular o menor caminho entres dois pontos do tabuleiro]
|   |   CatFather.py [estrutura genérica para ser herdada]
|   |
|   +---astarcat
|   |   |   AstarCat.py [gato que procura caminhos usando o algoritmo A*]
|   |   |   HeuristicValuesCalculator.py [calculo dos valores da função heurística]
|   |   |
|   +---catshelper [arquivos de utilidade para os gatos]
|   |   |   DistanceCalculator.py [cálculo de distâncias num grid hexagonal]
|   |   |   Path.py [representa o caminho entre dois pontos]
|   |
+---grid
|   |   Cell.py [representa uma casa do tabuleiro]
|   |   Grid.py [representa o tabuleiro com 11x11 casas]
|
\---util
    |   Helper.py [métodos comuns de ajuda]
```



Atribuição de scores para as casas do tabuleiro

O gato escolhe a casa com maior score

Cálculo do score envolve:

Média da distância dá casa para as três saídas mais próximas

Contagem de quantas vezes a distância mínima se repete em uma casa

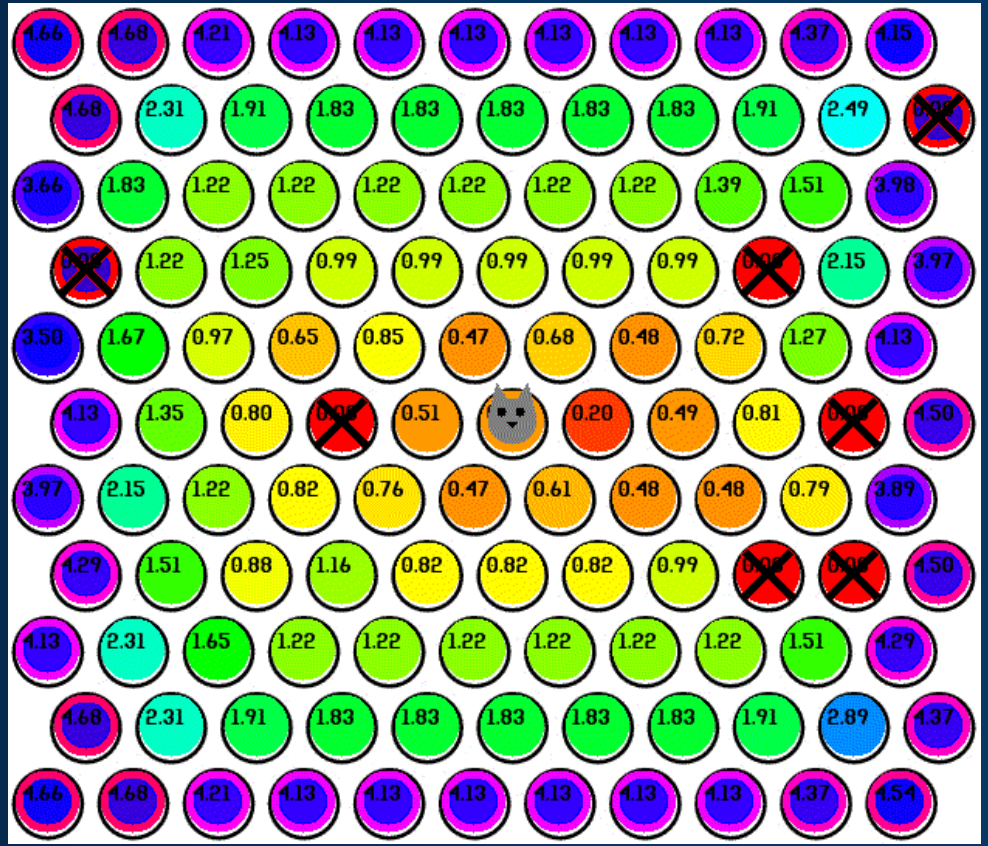
Se a casa é uma saída ou está do lado de uma

Comportamento do gato:

Naturalmente vai se afastar do centro indo para o lado com mais casas abertas

Uso de caminhos que confundam o pegador

Possibilidade de evitar casas próximas que serão fechadas antes do gato chegar caso os arredores tenham baixa score



```python
for elem, data in elems.items():

    s1_pathlen = CalculationHelper.normalize(
        data['path_len'],
        norm['path_len_min'],
        norm['path_len_max'],
        reversed=reverses[0],
    )

    s2_min_hits = CalculationHelper.normalize(
        data['min_hits'],
        norm['min_hits_min'],
        norm['min_hits_max'],
        reversed=reverses[1],
    )

    s5_near_goal = CalculationHelper.normalize(
        data['near_goal'],
        norm['near_goal_min'],
        norm['near_goal_max'],
        reversed=reverses[4],
    )

    score = (s5_near_goal * 3) + (s1_pathlen*2) + s2_min_hits


    if elem == (5,5):
        score *= 0.5

    if elem in [(5,4),(4,5),(4,6),(5,6),(6,6),(6,5)]:
        score *= 0.8
```

Normalização da média das três menores distâncias

Normalização da contagem de distâncias mínimas

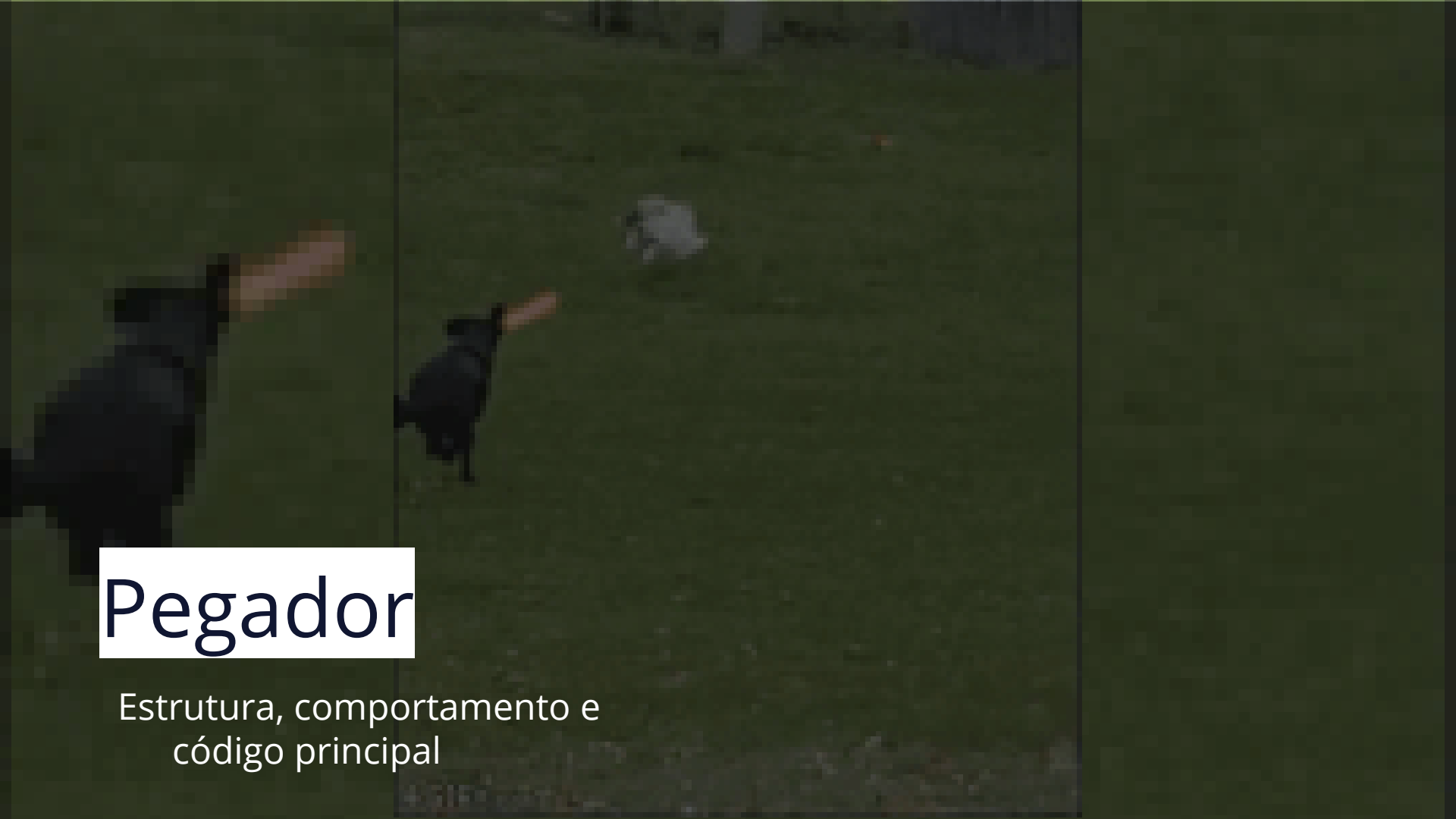
Normalização da nota pela proximidade da saída

A normalização usa um valor calculado para a casa e o máximo e mínimo desse valor para todas as casas, gerando um valor entre 0 e 1

Em alguns casos é preciso inverter esse valor – o score da distância foi invertido, *quanto menor a distância maior a score*

Cálculo do score

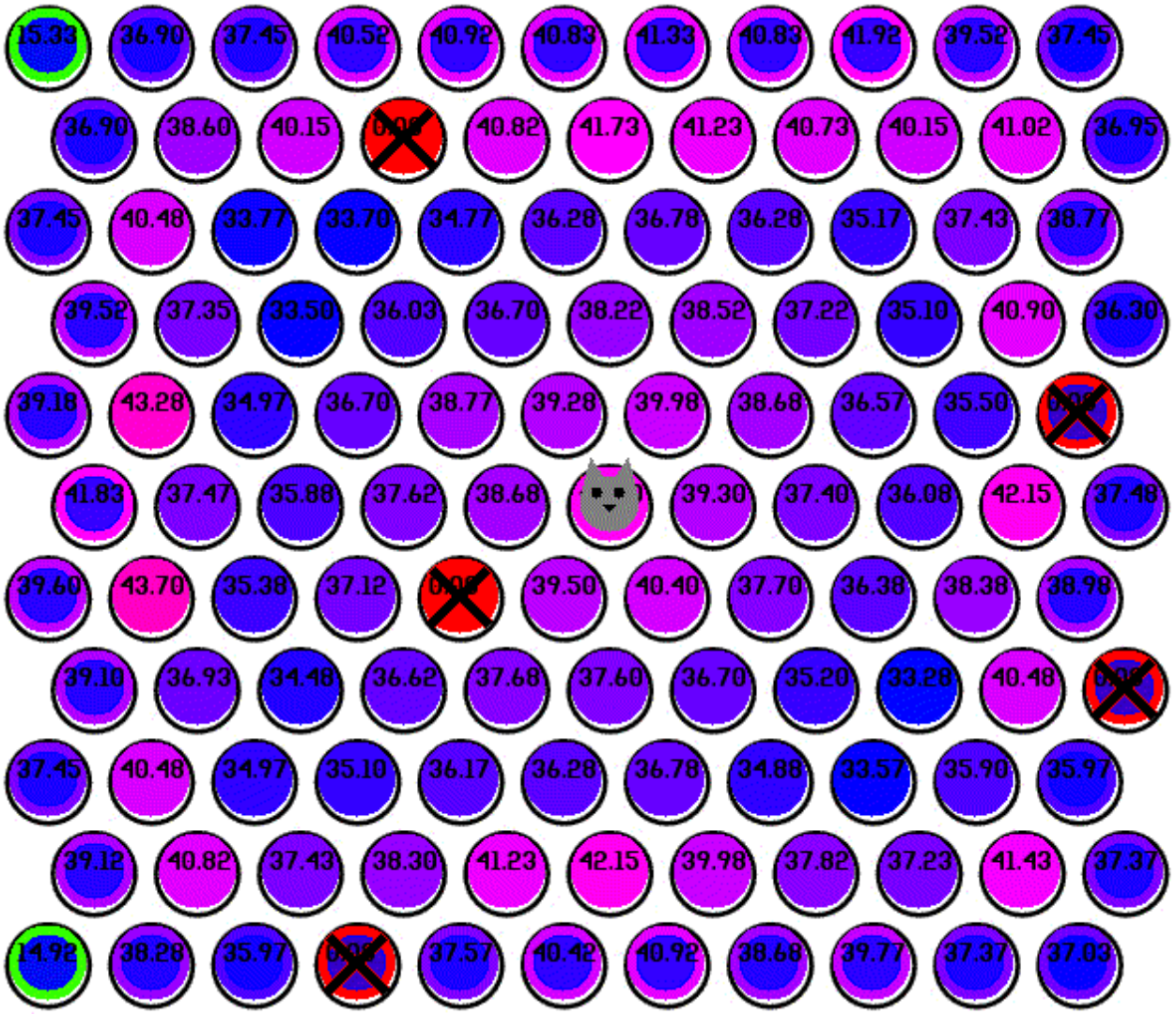
Diminuindo o score dos pontos centrais para evitar que o gato fique preso no início



Pegador

Estrutura, comportamento e código principal

```
root
|   catcher.py [arquivo que controla o pegador e a escolha do próximo passo]
|   scores_calculator_catcher.py [arquivo que atribui scores a pontos do tabuleiro]
|
+---cats [arquivos para calcular o menor caminho entres dois pontos do tabuleiro]
|   |   CatFather.py [estrutura genérica para ser herdada]
|   |
|   +---astarcat
|   |   |   AstarCat.py [gato que procura caminhos usando o algoritmo A*]
|   |   |   HeuristicValuesCalculator.py [calculo dos valores da função heurística]
|   |   |
|   +---catshelper [arquivos de utilidade para os gatos]
|   |   |   DistanceCalculator.py [cálculo de distâncias num grid hexagonal]
|   |   |   Path.py [representa o caminho entre dois pontos]
|   |
+---grid
|   |   Cell.py [representa uma casa do tabuleiro]
|   |   Grid.py [representa o tabuleiro com 11x11 casas]
|
\---util
    |   Helper.py [métodos comuns de ajuda]
```



Atribuição de scores para as casas do tabuleiro

O pegador fecha a casa com maior score

Cálculo do score envolve:

Distância da casa para o gato

A facilidade para o gato fugir

Casas perigosas

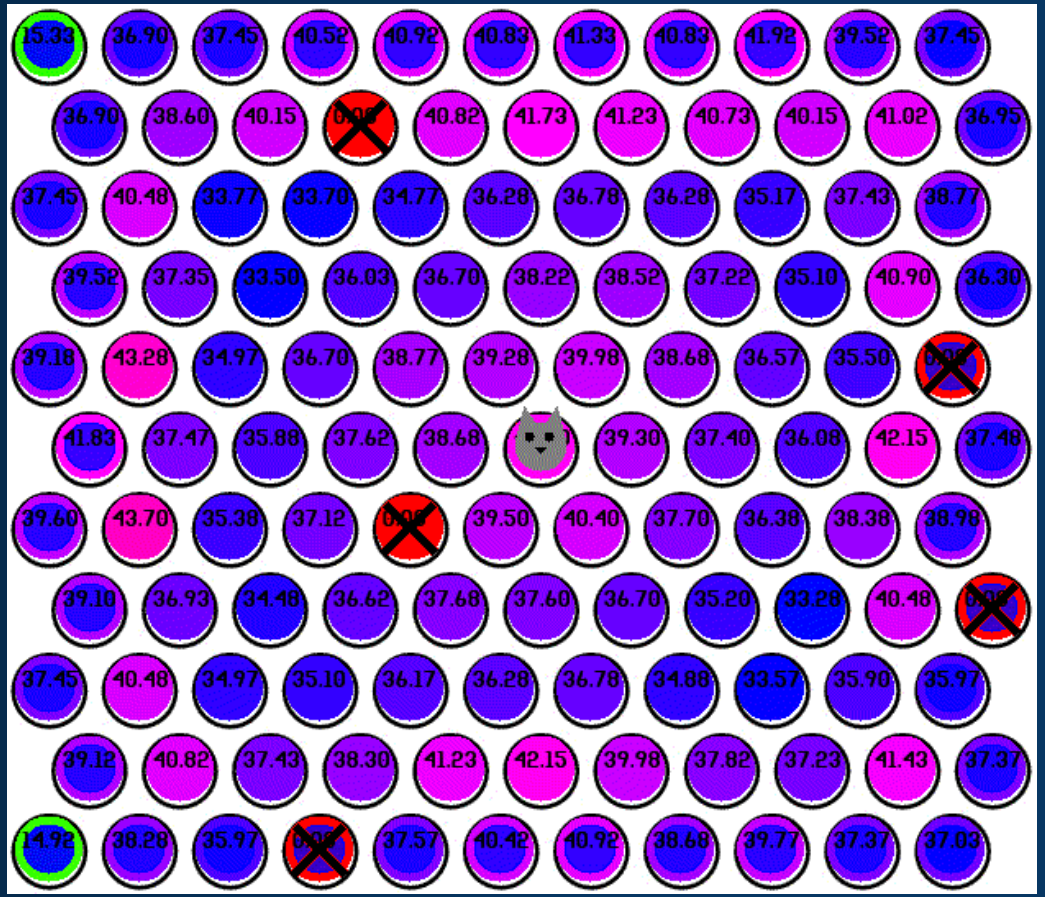
Casas interessantes de serem fechadas

Comportamento do pegador

De início se preocupa em fechar células em que caso o gato tenha acesso vai conseguir sair com certeza

Muda a estratégia caso o gato esteja muito perto de sair *(catcher panic)*

Usa estruturas pré-estabelecidas para prender o gato



Código do pegador

- Também foi utilizado a ideia de dar scores para as casas do tabuleiro
- Para casos com soluções mais simples métodos mais simples foram utilizados
- Divisão de diversas formas de pegar o gato em camadas ordenadas pela importância/capacidade do método
- Uso de uma lista com referências para os métodos a executar (na ordem da lista)
    - Caso um método retorne `None`, lance uma exceção ou não passe num teste de probabilidade o próximo da lista é o próximo a ser executado

Método a chamar

Nome dado a técnica do método (apenas para log)

Probabilidade de ser executado

```
def catch_them_all(self):
    solutions = list()

    solutions.append([self.catch_trapped_cat,               'catchTrapped',           1.0   ])
    solutions.append([self.block_cat_path,                  'block_cat_path',         0.1   ])
    solutions.append([self.catch_almost_trapped_cat,        'catchAlmostTrapped',     0.3   ])
    solutions.append([self.score_catcher,                   'score_catcher',          1.0   ])
    solutions.append([self.close_cat_is_near_exit_pathdist, 'catNearExitPathDist',    1.0   ])
    solutions.append([self.block_cat_before_goal_0,         'block_cat_before_goal_0', 1.0  ])
    solutions.append([self.block_exit_neib,                 'block_exit_neib',        1.0   ])
    solutions.append([self.mess_with_cat,                   'mess_with_cat',          1.0   ])
    solutions.append([self.close_stategic,                  'close_stategic',         1.0   ])
    solutions.append([self.heuristic_closer,                'heuristicCloser',        1.0   ])
    solutions.append([self.close_around_cat2,               'close_around_cat',       1.0   ])
    solutions.append([self.close_around_cat,                'close_around_cat',       1.0   ])
    solutions.append([self.generate_random_around_cat,      'randomAroundCat',        1.0   ])
    solutions.append([self.generate_random_position,        'closeRandom',            1.0   ])
```

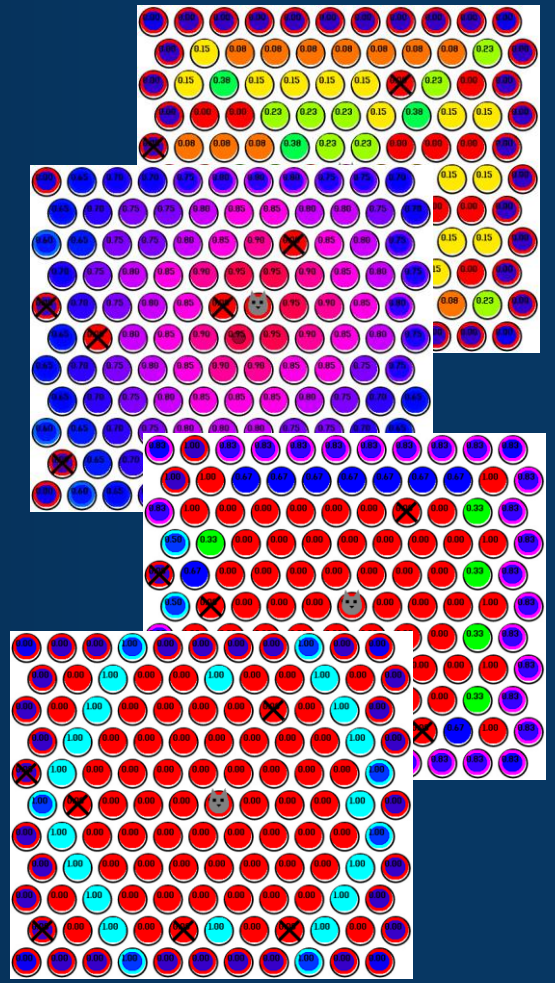
```
    s2_min_hits = Help.normalize(
        data['min_hits'],
        norm['min_hits_min'],
        norm['min_hits_max'],
        reversed=False,
    )

    s3_cat_dist = Help.normalize(
        data['cat_dist'],
        norm['cat_dist_min'],
        norm['cat_dist_max'],
        reversed=True,
    )

    s4_lead_to_goal = Help.normalize(
        data['leads_to_goals'],
        norm['leads_to_goals_min'],
        norm['leads_to_goals_max'],
        reversed=False,
    )

    s5_in_diamond = Help.normalize(
        data['in_diamond'],
        norm['in_diamond_min'],
        norm['in_diamond_max'],
        reversed=False,
    )
# ...
```

Normalização da contagem das menores distâncias

Normalização da distância para o gato

Normalização do score contando a quantas saídas uma casa fornece acesso

Score maior para certas casas formando um losango



```
s6_catcher_panic = Help.normalize(
    data['catcher_panic'],
    norm['catcher_panic_min'],
    norm['catcher_panic_max'],
    reversed=False,
)

s7_squares = Help.normalize(
    data['squares'],
    norm['squares_min'],
    norm['squares_max'],
    reversed=False,
)

s8_cat_trap = Help.normalize(
    data['cat_trap'],
    norm['cat_trap_min'],
    norm['cat_trap_max'],
    reversed=False,
)

# ...
```

Pegador entra em pânico quando o gato está muito perto de sair e altera as scores para bloquear o gato

Criação de uma série de quadros para prender o gato quando ele não puder mais sair

Caso o gato entre em um lugar onde não consiga sair a casa que o prende tem score maior