ISSN 2177-9139

XX EREMAT - Encontro Regional de Estudantes de Matemática da Região Sul
Fundação Universidade Federal do Pampa (UNIPAMPA), Bagé/RS, Brasil. 13-16 nov. 2014.

XX EREMAT SUL

Encontro Regional de Estudantes de Matemática da Região Sul

UM MODELO DE SIMULAÇÃO A PARTIR DO AMBIENTE NETLOGO

Celso Nobre da Fonseca- e-mail: celsonf@gmail.com.
Professor da Universidade Católica de Pelotas (UCPel) - Brasil, Rio Grande do Sul, Pelotas, Centro, Rua Gonçalves Chaves, 373, CEP 96015-560.
Rosana dos Santos - e-mail: profrosanasantos@gmail.com.
Professora/Tutora à distância UAB/UFPel - Universidade Federal de Pelotas (UFPel) - Brasil, Rio Grande do Sul, Capão do Leão, Campus Universitário, S/N - CEP 96160-000.

Resumo. *Esta oficina tem por objetivo apresentar o ambiente NetLogo como uma ferramenta de auxílio didático - pedagógico, suas características, especificações e aplicações que podem ser modeladas e implementadas em tal ambiente, já que o mesmo possui uma vasta biblioteca de modelos. Além disto, será apresentado um modelo de simulação de agentes a partir do ambiente NetLogo, no intuito de que se possa assimilar alguns dos comandos e primitivas que compõem as simulações do mesmo.*

Palavras - chave: NetLogo, Agentes, Modelos de simulação.

1. INTRODUÇÃO

A linguagem Logo foi criada por Seymour Papert e Marvin Minski, pesquisadores do grupo de pesquisa Logo, do Laboratório de Inteligência Artificial do Instituto de Tecnologia de Massachusetts (MIT), no final dos anos setenta, a partir do projeto "uma teoria geral da inteligência" ("A Teoria da Sociedade da Mente"). O projeto emergiu de uma estratégia de refletir ao mesmo tempo sobre como as crianças pensam e como os computadores poderiam "pensar" (PAPERT, 1985- p. 244).

De acordo com SEYMOUR(2001- p. 4), o Logo é uma linguagem computacional concebida para crianças. Ela privilegia a interatividade através do controle de uma tartaruga cibernética que é, por definição, "burra" por uma criança "inteligente".

O NetLogo, no entanto, é uma plataforma de simples programação que tem como propósito principal fornecer ao programador as ferramentas básicas necessárias para se poder dar ações a determinadas entidades, chamadas de agentes, afim de que estes possam realizar ações dentro de um ambiente virtual.

Nesta oficina será apresentado aspectos gerais do ambiente NetLogo, ressaltando o uso das principais primitivas que podem ser usadas para a realização de diversos tipos de simulações.

2. AMBIENTAÇÃO DO NETLOGO

Um tipo de simulação de grande importância quando se trata de comportamentos complexos é a chamada simulação baseada em agentes. Um agente, no entanto, pode ser colocado como toda a entidade, real ou virtual, que tem como principal característica a tomada de decisões conforme a mudança de estado de um determinado ambiente.

Com o desenvolvimento dos computadores e softwares, diversos ambientes voltados a realizarem simulações de agentes, que visavam fazer estudos de determinados comportamentos foram desenvolvidos, sendo um destes o NetLogo, o qual permite ao usuário explorar suas ferramentas afim de desenvolver simulações que visam estudar comportamentos de sistemas complexos, seja estes visual ou estatístico.

O corpo do NetLogo é composto de três grupos de agentes nos quais se podem atribuir comportamentos, sendo eles o *observer*, os *patches*, e as *turtles*. O *observer* é o grupo de agentes que irão especificar e dar as condições de funcionamento de cada objeto presente na simulação. Os *patches* especificam e dão condições de funcionamento somente dos objetos que estão associados aos agentes imóveis do NetLogo, enquanto *turltes* somente aos agentes móveis.

O programador pode dar simultaneamente no NetLogo várias instruções a milhares de agentes independentemente, tornando possível explorar os comportamentos que aparecem quando um determinado número destes interagem entre si.

A interface do NetLogo traz consigo algumas ferramentas exploratórias que são, o *file*, o *edit*, *tools*, *zoom* , *tabs* e *help*, os quais dão as diferentes ferramentas de edição do ambiente e dos agentes.

2.1 Interface e principais características

Ao inicializar a ferramenta NetLogo é possível visualizar alguns objetos relevantes. O primeiro deles é um espaço escuro a direita, que é o lugar reservado para visualizar a ação dos diferentes agentes. O segundo são as ferramentas exploratórias existentes na parte superior conforme pode ser visto na Figura 2.1.1.

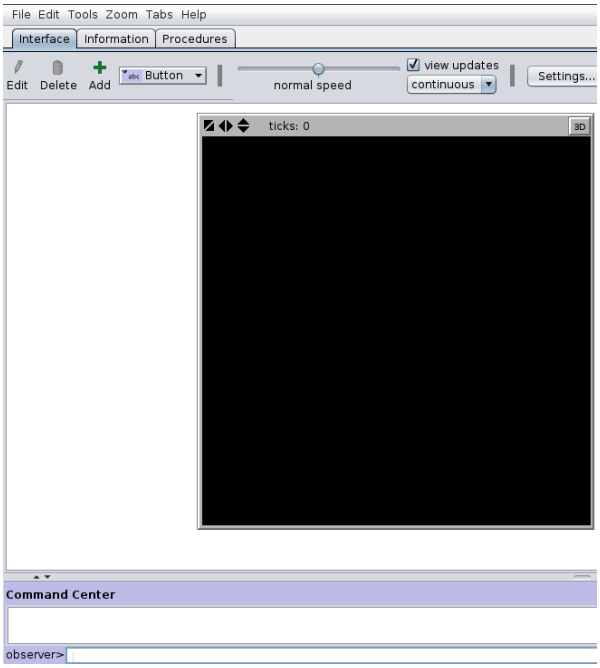

Figura 2.1.1: Interface do ambiente NetLogo.

O espaço em branco a esquerda na interface (Figura 2.1.1), e ao lado do espaço onde se pode visualizar os agentes, serve para adicionar alguns botões gráficos, que podem ser utilizados principalmente para o controle manual ou para a visualização de algum comportamento conforme assim deseje o programador.

Tem-se os botões de controle manual definidos como *button*, *slider*, *switch*, *chooser* (Figura 2.1.2), e os botões *monitor*, *plot*e *output*que são para a visualização do comportamento de variáveis (Figura 2.1.3).

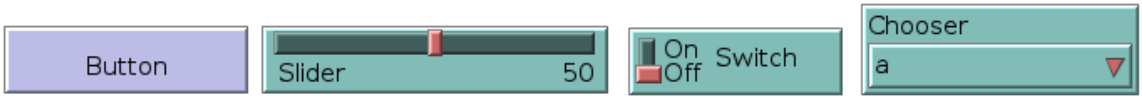

Figura 2.1.2: Botões *button*, *slider*, *switch* e *chooser*.

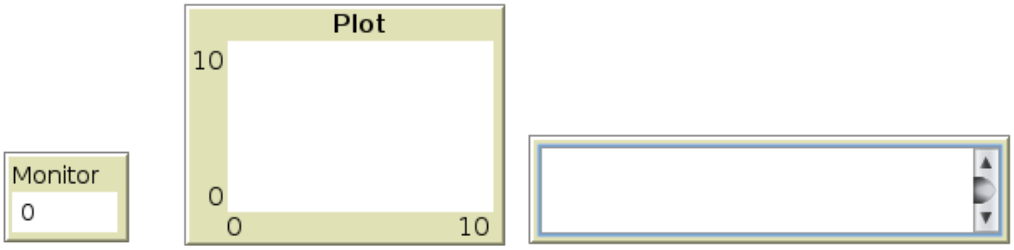

Figura 2.1.3: Botões *monitor*, *plot e output.*

2.2 Agentes do NetLogo

Quando um agente é criado no NetLogo a sua posição pode ser definida conforme desejado. Na figura abaixo (Figura 2.2.1), o agente está com a "cabeça" apontada numa direção de 270º em relação a parte superior da tela. Vale salientar que o NetLogo toma a direção 0º como sendo a parte superior da tela, 90º a parte direita, 180º a parte inferior, e 270º a parte esquerda da tela, em relação ao agente.

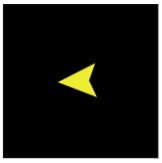

Figura 2.2.1: Posição do agente no NetLogo.

Na figura anterior (Figura 2.2.1) pode-se ver um agente de cor amarela e na forma de uma seta. O NetLogo, quando não se define uma forma (quadrado, círculo, gato, rato, etc.) o NetLogo toma os agentes sempre como sendo setas.

2.3 Variáveis próprias dos agentes

Todo agente criado no NetLogo carrega consigo determinadas variáveis que servem como características para identificar sua posição em relação ao ambiente e suas peculiaridades. Para visualizá-las, posiciona-se a seta do mouse sobre o agente, clica-se com o botão da direita e depois em *inspect turtle*. As variáveis são: o *who, color, heading, xcor, ycor, shape, label,*

label-color, *breed*, *hidden?*, *size*, *pen-size* e *pen-mode*, conforme a figura a seguir (Figura 2.3.1).

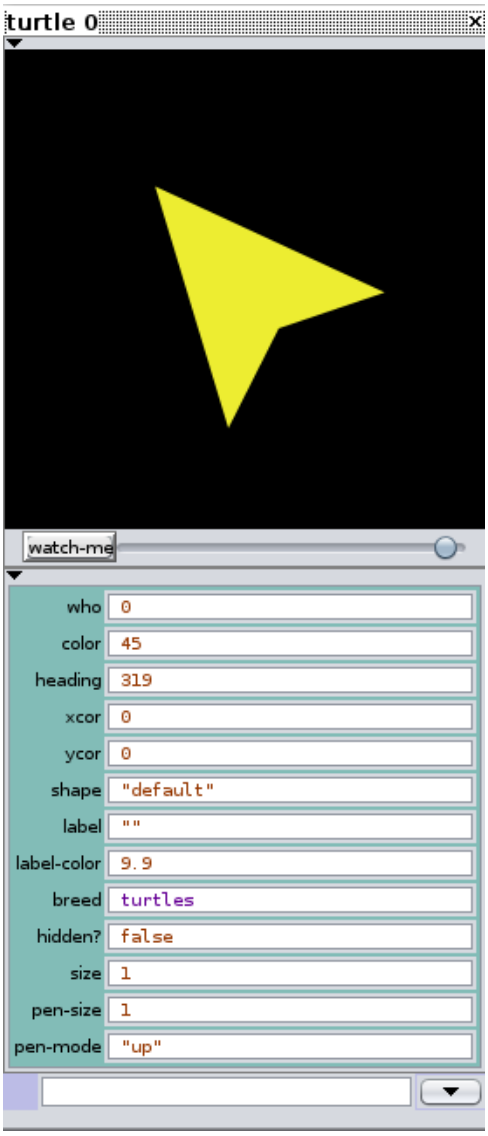

Figura 2.3.1: Variáveis próprias dos agentes.

2.4 Programação de agentes no NetLogo

No ambiente NetLogo há duas formas de programar, sendo uma delas diretamente através do *command center*,escolhendo um dos grupos, ou através do *procedures*. No primeiro, determina-se um comando e visualiza-se a execução diretamente, porém neste, a desvantagem é que o comando será executado uma única vez, o que pode ser desgastante para grandes simulações que necessitam de vários comandos e procedimentos simultâneos. Já no *procedures*, podem-se colocar muitos comandos e procedimentos, e utilizá-los sempre que necessários. Clicando sobre a barra *procedures* do NetLogo é possível visualizar uma janela em branco, que é o local onde os procedimentos e ações dos agentes podem ser programados.

2.5 Primitivas do NetLogo

O NetLogo possui uma coleção de primitivas já prontas, as quais podem ser combinadas, de maneira coerente, de forma a possibilitar programar as diferentes ações que se deseja a um único, ou a um conjunto inteiro de agentes. Como os principais agentes no NetLogo são de dois tipos, *turtles* e *patches*, há uma gama de primitivas para ambos, e se identificam como *Turtle-related* e *Patch-related*. Além disto, o NetLogo também possui um conjunto de primitivas que podem ser usadas tanto para as *turtles* como para os *patches*, os chamados *Agentset*.

As primitivas do *Turtle-related* servem para dar instruções aos agentes móveis, como por exemplo, a maneira que devem se mover ou como interagir com outros agentes. As primitivas do *Patch-related* dão instruções aos agentes imóveis, como por exemplo, qual deve ser sua cor ou contar quantas *turtles* estão sobre o *patch*es, etc.

3. CRIANDO UM AMBIENTE NO NETLOGO

Nesta seção será exemplificado como programar o ambiente e os agentes, afim de que se possa assimilar alguns dos comandos e primitivas que compõem as simulações no NetLogo. No exemplo "gato-caça-rato", será ilustrado como trabalhar com o NetLogo de acordo com a idéia de ver o gato, ou gatos, caçando o rato, ou ratos.

Inicialmente, como em qualquer tipo de simulação, deve-se ter em mente qual, ou quais os tipos de agentes e seus comportamentos que fazem parte da simulação e neste exemplo, os agentes tem duas características bem distintas, ou são gatos, ou são ratos, assim como dois tipos de comportamentos, que são fugir (ratos) e caçar (gatos).

Na tela *procedures* do NetLogo será construído dois tipos de agentes com a primitiva *breed* que são os gatos e os ratos conforme figura abaixo (Figura 3.1).

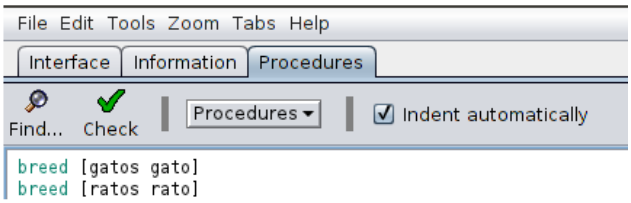

Figura 3.1: Primitiva *breed.*

A primitiva *breed* recebe dois argumentos, um no plural, e outro no singular. O argumento no plural servirá para dar características a todos agentes com o mesmo no nome, no caso gatos e ratos, enquanto o argumento no singular servirá para dar características a um único agente de nome gato ou rato. Deve ser lembrado que toda a ação ou procedimento no NetLogo deve começar com um *to* e terminar por um *end*.

Nesta simulação, escolheu-se optar por dois nomes bem distintos, o *setup* e o *go*. O *setup* é o procedimento que irá colocar as principais características do ambiente, como por exemplo, quantidades de ratos e de gatos inicialmente na simulação. O *go* é o procedimento que irá colocar todas as ações que determinam o que os agentes devem fazer na simulação, como por exemplo, como andar, caçar, morrer, entre outros. Dentro do procedimento *setup* criou-se os procedimentos "criar-gatos" e "criar-ratos", que nesta simulação será inicialmente de 10 gatos e 20 ratos.

Dentro do procedimento "criar-gatos" foram colocadas as seguintes instruções: gato de cor azul - "*blue"*; forma de gato - *"cat"* e posições aleatórias - "*set xcor random-xcor"* e *"set ycor random-ycor"*. No procedimento "criar-ratos" foram colocadas as instruções: ratos de cor vermelha - "*red*" e forma de rato - *"mouse top"*.

Para visualizar como será dada a inicialização da simulação no ambiente colocasse um botão na interface, e no espaço abaixo de *commands* escreve-se *setup* conforme a figura abaixo (Figura 3.2).

Ao clicar no botão *setup* a tela escura mostrará os diferentes agentes que foram definidos anteriormente (Figura 3.3).

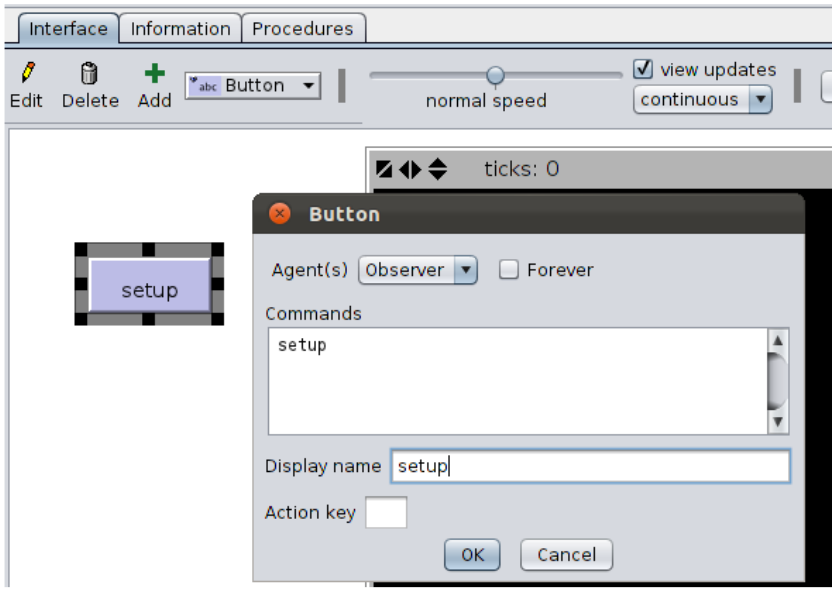

Figura 3.2: Botão com procedimento *setup*.

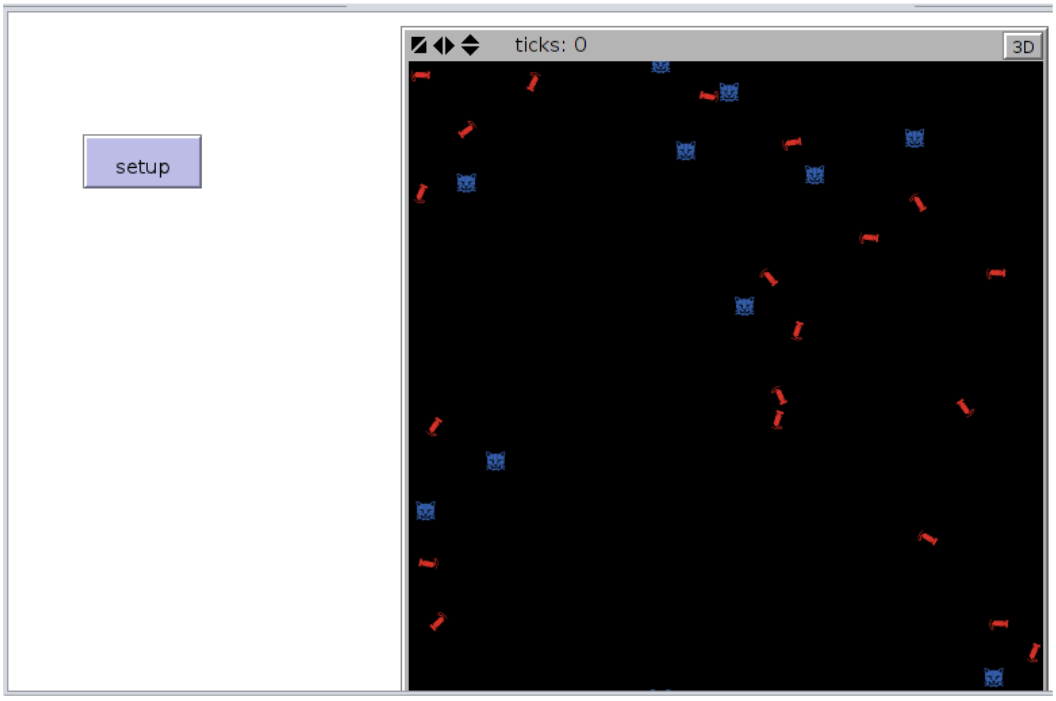

Figura 3.3: Agentes gatos e ratos.

Os procedimentos para a simulação anterior são os descritos a abaixo.

```
to setup   ; NOME DADO AO PROCEDIMENTO QUE IRÁ INICIALIZAR O AMBIENTE
  ca
  criar-gatos
  criar-ratos
end

to criar-gatos
  create-gatos 1
           [
             set color blue
             set shape "cat"
             set xcor random-xcor
             set ycor random-ycor
           ]
end
to criar-ratos
  create-ratos 1
           [
             set color red
             set shape "mouse top"
             set xcor random-xcor
             set ycor random-ycor
           ]
end
```

No modelo anterior foi definido como saber quantos ratos um determinado gato conseguiria caçar. Para tanto definiu-se uma variável somente para o agente gato cujo nome da variável foi "*qtd-ratos"*. Para se criar uma variável própria de um determinado agente, seja ela qual for, na tela *procedures* digita-se o nome do agente juntamente com a primitiva *own*. Ao clicar no botão *setup* e com o botão direito do mouse em cima de um dos agentes *gato* será visualizado a nova variável *qtd-ratos* conforme a figura abaixo (Figura 3.4).

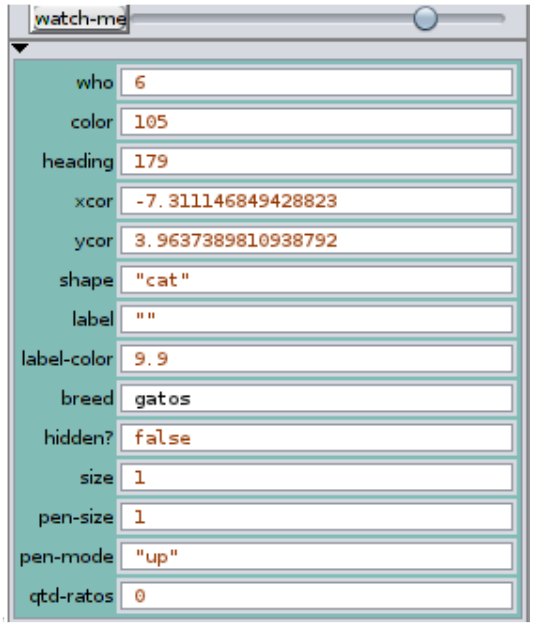

Figura 3.4: Variável *qtd-ratos.*

3.1. Ações dos gatos e dos ratos

Tanto os gatos quanto os ratos irão se movimentar no ambiente, e para que esta simulação se pareça mais próximo dos padrões naturais de caça e fuga, será adicionado alguns procedimentos que os farão mais inteligentes em relação a caça e a fuga.

Para o movimento dos agentes deve-se adicionar um procedimento que fará tanto ratos como gatos se movimentarem no ambiente (*patches*). Para isto adiciona-se o procedimento que será chamado de "movimentar-agente" e onde serão utilizados as primitivas *ask*, *fd* e *set heading* que serão adicionadas as ações *go*. Os procedimentos são os seguintes:

```
to go   ;NOME DADO AO PROCEDIMENTO QUE IRÁ REALIZAR AS AÇÕES

    movimentar-agente

end
to movimentar-agente
  ask turtles
     [
       fd 1
       set heading random 360
     ]
end
```

Neste caso o agente irá se movimentar um passo (*fd 1*) e girar em torno de seu eixo um ângulo randômico que vai de 0º até 360º (*set heading random 360*).

3.2 Ações de caça do gato e fuga do rato

Para que na simulação o gato seja um pouco mais esperto, deve-se supor que os gatos só corram atrás daqueles ratos que estejam mais perto dele. Para este caso, pode-se tomar que a distância mínima a que um rato deve estar do rato para que o mesmo comece a segui-lo seja de 5 *patches*. Para isto, desenvolveu-se o seguinte procedimento e o acrescentou-se a ação *go:*

```
to go   ;NOME DADO AO PROCEDIMENTO QUE IRÁ REALIZAR AS AÇÕES

    movimentar-agente
    gato-caca-rato
end
to movimentar-agente
  ask turtles
     [
       fd 1
       set heading random 360
     ]
end

to gato-caca-rato
   ask ratos
     [
       let distancia self
       ask gatos with[distance distancia < 5]
         [
           face distancia
         ]
     ]
end
```

Nesta ação, no entanto, o gato somente irá perseguir o rato que estiver com uma distância (*distance* < *5*), mas nada irá acontecer ao rato se não adicionarmos ao procedimento as seguintes linhas:

```
to gato-caca-rato
   ask ratos
     [
       let distancia self
       ask gatos with[distance distancia < 5]
         [
           face distancia
         ]

     ]
    ask gatos
         [
           ask ratos-here
             [
               die
             ]
         ]
end
```

A primitiva *face distance* faz neste caso com que o gato persiga o rato, já a primitiva *ask ratos-here* [*die*] faz com que o rato que estiver no mesmo lugar do gato morra (*die*).
Veja que nas linhas do procedimento anterior, o rato não tem chance de escapar do gato, pois o rato não tem nenhuma ação até então, a qual possa ao menos dar uma chance de sobrevivência. Logo, para que o rato tenha uma chance de sobrevivência perante a quantidade de gatos, foi suposto que ao ver o gato o rato tente mudar sua direção girando 180º. Para isto adicionou-se o seguinte código a ação *go* conforme consta a seguir:

```
to go   ;NOME DADO AO PROCEDIMENTO QUE IRÁ REALIZAR AS AÇÕES

    movimentar-agente
    gato-caca-rato
    rato-esperto
end

to rato-esperto
  ask ratos
    [
      if any? gatos in-cone 3 60 = true
        [
          set heading heading + 180
          fd 2
        ]
    ]
end
```

No trecho de código adicionou-se o teste *if* e foi dito ao rato, caso ele enxergue um gato num raio de 3 *patches* e ângulo de visão de 60º (*in-cone 3 60*) vire sobre o seu eixo um ângulo de 180º (*set heading heading + 180*).

Além do comando anterior, pode-se adicionar facilitar a vida do rato adicionando uma chance de 40% de escapar do gato no procedimento “gato-caca-rato”, mesmo o gato estando em cima do rato, conforme o procedimento abaixo.

```
to gato-caca-rato
   ask ratos
     [
       let distancia self
       ask gatos with[distance distancia < 5]
         [
           face distancia
         ]

     ]
    ask gatos
         [
           ask ratos-here
             [
               if random 100 > 60
                [
                  ask gatos-here
                    [
                      set qtd-ratos qtd-ratos + 1
                    ]
                  die

                ]
             ]
         ]
end
```

Vale lembrar que neste último trecho já está adicionado o contador de ratos de cada gato *set qtd-rato qtd-rato + 1*.

Para finalizar a simulação adiciona-se o botão com o nome *go* e com a opção *forever* conforme a figura a seguir (Figura 3.2.1).

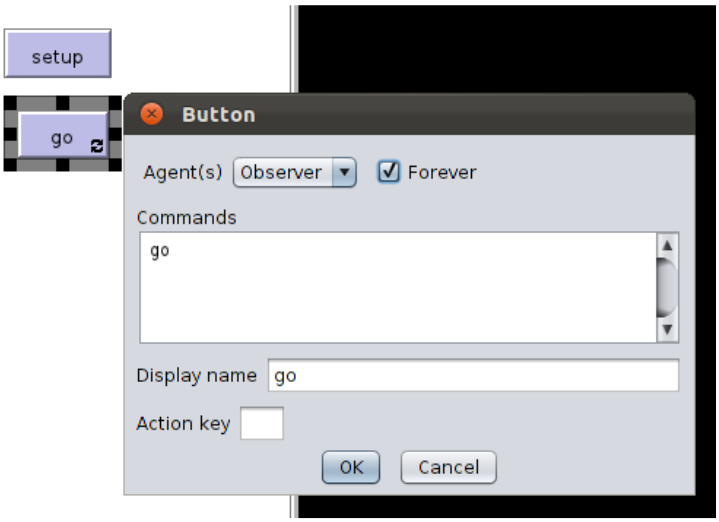

Figura 3.2.1: Botão de ação *go.*

Finalmente, para realizar a simulação somente é necessário clicar sobre o botão *setup* e depois o botão *go*.

4. CONSIDERAÇÕES FINAIS

O ambiente NetLogo possui uma multiplicidade de modelos de diferentes domínios, e que pode ser utilizado em uma larga variedade de contextos educacionais, desde a escola elementar até a graduação.

A simulação vista no decorrer do material é uma de milhares que podem ser trabalhadas com tal ambiente, como por exemplo, modelos ligados a economia, biologia, física, matemática, química, psicologia, meteorologia, medicina, veterinária, entre outros.

Espera-se que a partir dos comandos básicos apresentados neste material, seja mais compreensível o uso do ambiente NetLogo, tanto para os estudantes, professores, como para pesquisadores de diferentes áreas do conhecimento.

5. REFERÊNCIAS

LIMA, T. F. M.; FARIA, S. D.; FILHO, B. S. S.; CARNEIRO, T. G. S. **Modelagem de sistemas baseada em agentes: alguns conceitos e ferramentas**. XIV Simpósio Brasileiro de Sensoriamento Remoto, Natal, Brasil, 25-30 abril 2009, INPE, p. 5279-5286.

LOBÃO, E. C.; PORTO, A. J. V. **Evolução das técnicas de simulação**. Escola de engenharia de São Carlos- EESC. São Paulo, Brasil, 1999.

NIGEL G. **Agent-based models**.University of Surrey, Guildford, UK, 2011.

NIGEL, G., Troitzsch, K.G. **Simulation for the social scientist**. University of Surrey, Guildford, UK, 1999.

NIGEL, G. **Agent-based social simulation: dealing with complexity**. University of Surrey, Guildford, UK, 2004.

PANTAROLO, E.; AZEVEDO, L. L.; MENEZES, C. S.; MAGDALENA B. C. **Exploração do ambiente orientado a agente NetLogo**. XVI Simpósio Brasileiro de informática na educação. Juiz de Fora, Minas Gerais, Brasil, 2005.

RUSSEL, S.; NORVIG, P. **Inteligência artificial**. Editora Campus, São Paulo, Brasil, 2003.

PAPERT, Seymour. **Logo: Computadores e educação**. São Paulo: Brasiliense, 1985.