Fundação CECIERJ – Vice Presidência de Educação Superior à Distância
Curso de Tecnologia em Sistemas de Computação
Disciplina: Programação II
Gabarito da AD2 – 1º Semestre de 2013

Questão 1

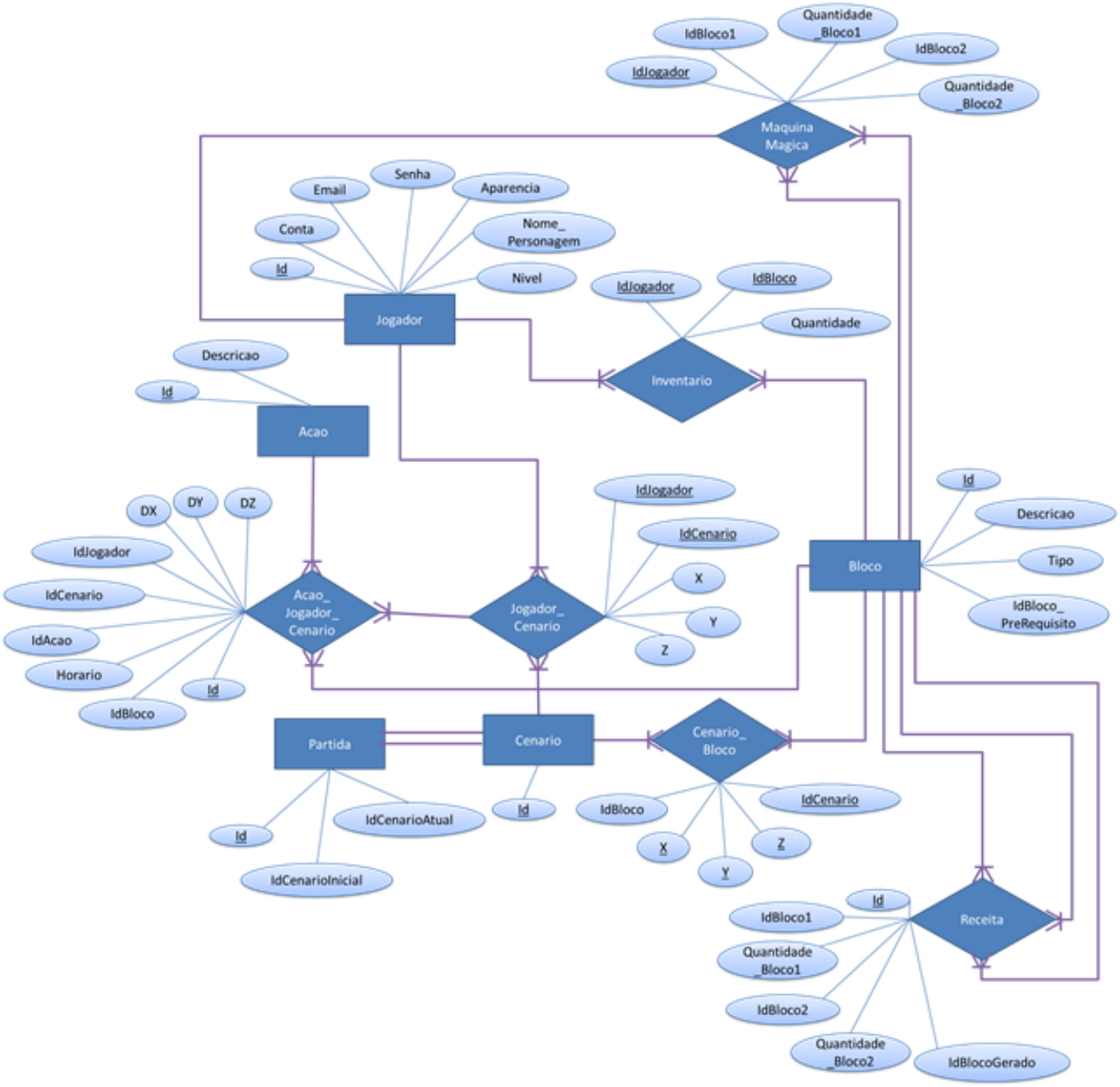
Deseja-se modelar um jogo de computador, apelidado BitBriks, onde seu personagem, constituído por um boneco de baixa resolução, interage com um mundo tridimensional através do movimento de blocos do cenário e da transformação destes blocos em acessórios ou blocos diferentes. O jogo precisará de um banco de dados que permita a interação de jogadores (multiplayer), e deve atender às seguintes características:

- Para cada jogador, manter a conta, senha, email, nome do personagem, inventário contendo seus itens, nível do personagem, aparência escolhida;
- O mapeamento do cenário da partida, que é basicamente constituído por blocos cúbicos de mesma dimensão e certo tipo, espalhados em posições de um eixo coordenado tridimensional. Cada personagem possui sua posição inicial definida e também possui as dimensões de um bloco, não havendo dessa forma a possibilidade de superposição entre blocos/personagens no mesmo lugar do espaço; Devem ser guardados o cenário inicial, e o cenário atual, após a última rodada de ações;
- Qualquer bloco do cenário ao qual um personagem esteja tangente pode ser trazido para o inventário. Quando mais de um bloco de um mesmo tipo está no inventário, eles são então agrupados, formando um indicador numérico, que pode atingir no máximo o valor noventa e nove. Existem blocos que possuem como pré-requisito que o usuário possua determinado acessório em seu inventário, e do contrário, não serão colhidos;
- Cada jogador deve ter seu histórico de ações rastreado, na forma de um par <ação,direção>, sendo possível ver o *replay* da partida, munido somente deste histórico e da posição inicial dos jogadores. As ações são fracamente acopladas, podendo ser expandidas em atualizações, e são como exemplo: andar, pegar bloco, largar bloco selecionado. As direções podem ser quaisquer das 27 posições adjacentes, incluindo a própria posição, indicando que o personagem permanecerá parado para esta ação. Nos casos em que um elemento está selecionado, a informação sobre este elemento também deve ser armazenada;

- Cada jogador possui uma máquina mágica, na qual poderá executar receitas, capazes de receber até dois tipos de blocos em determinada quantidade e transformá-los, ação especial chamada de “transformar”. Caso seja colocado na máquina apenas 1 tipo de bloco em quantidade suficiente, o resultado será a geração de um novo tipo de bloco e o retorno do material excedente. No caso de dois tipos de bloco diferente, também em quantidades determinadas, o resultado será a geração de um acessório, que pode ser usado para facilitar sua interação com o mundo, abrindo ao jogador um leque maior de tipos de ações. Os acessórios não podem ser transformados ou jogados fora;
- O jogo não possui Inteligência Artificial, ou seja, apenas jogadores e cenário. Tampouco há um critério de parada, vitória, etc.

a) Faça a modelagem Entidade-Relacionamento de um banco de dados capaz de atender as necessidades do jogo BitBriks. Preze por simplicidade, ao passo em que não deve deixar funcionalidades explicitadas na descrição da questão não atendidas por seu modelo.

R: O diagrama a seguir representa uma das possíveis soluções para o problema. Não existe uma única resposta certa para o enunciado, sendo também consideradas corretas soluções que não representarem os elementos Partida e Maquina_Magica, mas que consigam modelar o banco de forma que constem as informações pedidas na questão.



b) Escreva também o script de criação de tabelas (modelo físico) do problema.

R:

```
CREATE TABLE `cenario`
  (
     `id` INT NOT NULL auto_increment,
     PRIMARY KEY (`id`)
  );
```

```sql
CREATE TABLE `partida`
  (
     `id`               INT NOT NULL auto_increment,
     `idcenarioinicial` INT NOT NULL,
     `idcenarioatual`   INT NOT NULL,
     PRIMARY KEY (`id`),
     FOREIGN KEY (`idcenarioinicial`) REFERENCES `cenario` (`id`),
     FOREIGN KEY (`idcenarioatual`) REFERENCES `cenario` (`id`)
  );

CREATE TABLE `bloco`
  (
     `id`                   INT NOT NULL auto_increment,
     `descricao`            VARCHAR(255) NOT NULL,
     `tipo`                 ENUM('material', 'acessorio') NOT NULL,
     `idbloco_prerequisito` INT DEFAULT NULL,
     PRIMARY KEY (`id`)
  );

ALTER TABLE `bloco`
  ADD FOREIGN KEY (`idbloco_prerequisito`) REFERENCES `bloco`(`id`);

CREATE TABLE `cenario_bloco`
  (
     `idcenario` INT NOT NULL,
     `idbloco`   INT NOT NULL,
     `x`         INT DEFAULT NULL,
     `y`         INT DEFAULT NULL,
     `z`         INT DEFAULT NULL,
     PRIMARY KEY (`idcenario`, `x`, `y`, `z`),
     FOREIGN KEY (`idbloco`) REFERENCES `bloco` (`id`),
     FOREIGN KEY (`idcenario`) REFERENCES `cenario` (`id`)
  );

CREATE TABLE `acao`
  (
     `id`        INT NOT NULL auto_increment,
     `descricao` VARCHAR(255) NOT NULL,
     PRIMARY KEY (`id`)
  );
```

```sql
CREATE TABLE `jogador`
  (
     `id`              INT NOT NULL auto_increment,
     `conta`           VARCHAR(50) NOT NULL,
     `email`           VARCHAR(255) NOT NULL,
     `senha`           CHAR(40) NOT NULL,
     `aparencia`       INT NOT NULL,
     `nome_personagem` VARCHAR(255) NOT NULL,
     `nivel`           INT NOT NULL DEFAULT '1',
     PRIMARY KEY (`id`)
  );

CREATE TABLE `jogador_cenario`
  (
     `idjogador` INT NOT NULL,
     `idcenario` INT NOT NULL,
     `x`         INT NOT NULL,
     `y`         INT NOT NULL,
     `z`         INT NOT NULL,
     PRIMARY KEY (`idjogador`, `idcenario`),
     FOREIGN KEY (`idjogador`) REFERENCES `jogador` (`id`),
     FOREIGN KEY (`idcenario`) REFERENCES `cenario` (`id`)
  );

CREATE TABLE `acao_jogador_cenario`
  (
     `id`        INT NOT NULL auto_increment,
     `idjogador` INT NOT NULL,
     `idcenario` INT NOT NULL,
     `idacao`    INT NOT NULL,
     `idbloco`   INT,
     `horario`   TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
     `dx`        INT,
     `dy`        INT,
     `dz`        INT,
     PRIMARY KEY (`id`),
     FOREIGN KEY (`idjogador`, `idcenario`) REFERENCES
`jogador_cenario`(`idjogador`, `idcenario`),
     FOREIGN KEY (`idacao`) REFERENCES `acao`(`id`),
     FOREIGN KEY (`idbloco`) REFERENCES `bloco`(`id`)
  );
```

```sql
CREATE TABLE `inventario`
  (
     `idjogador`  INT NOT NULL,
     `idbloco`    INT NOT NULL,
     `quantidade` INT DEFAULT 1,
     PRIMARY KEY (`idjogador`, `idbloco`),
     FOREIGN KEY (`idjogador`) REFERENCES `jogador`(`id`),
     FOREIGN KEY (`idbloco`) REFERENCES `bloco`(`id`)
  );

CREATE TABLE `maquinamagica`
  (
     `idjogador`        INT NOT NULL,
     `idbloco1`         INT,
     `quantidadebloco1` INT,
     `idbloco2`         INT,
     `quantidadebloco2` INT,
     PRIMARY KEY (`idjogador`),
     FOREIGN KEY (`idbloco1`) REFERENCES `bloco`(`id`),
     FOREIGN KEY (`idbloco2`) REFERENCES `bloco`(`id`)
  );

CREATE TABLE `receita`
  (
     `id`                INT NOT NULL,
     `idbloco1`          INT NOT NULL,
     `idbloco2`          INT,
     `quantidade_bloco1` INT NOT NULL,
     `quantidade_bloco2` INT,
     `idblocogerado`     INT NOT NULL,
     PRIMARY KEY (`id`),
     FOREIGN KEY (`idbloco1`) REFERENCES `bloco`(`id`),
     FOREIGN KEY (`idbloco2`) REFERENCES `bloco`(`id`),
     FOREIGN KEY (`idblocogerado`) REFERENCES `bloco`(`id`)
  );
```

Questão 2

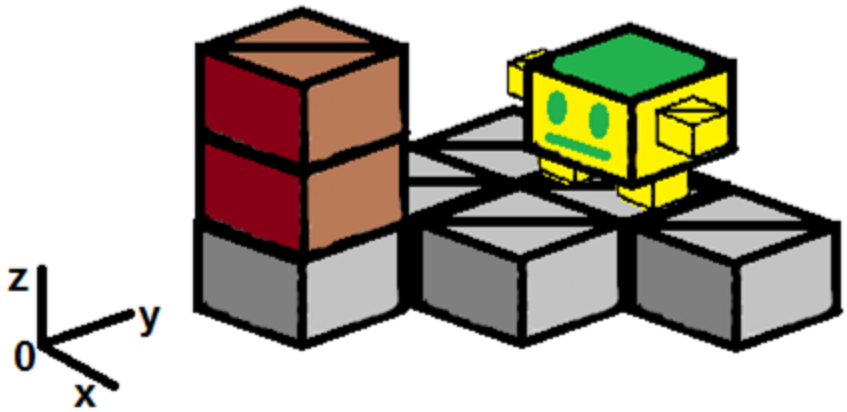
Baseado na descrição do BitBriks, escreva a função executar_acao($acao, $direcao, $ref_obj=null), que receba como parâmetros obrigatórios: a referência da ação, a direção a ser tomada, e se existir, a referência para um bloco/acessório envolvido na ação. O script deve:

- Pré-popular todas as tabelas envolvidas, inclusive a de receitas e um cenário hipotético,

e fazer 5 chamadas a função executar_acao, abrangendo todas as possíveis ações (Os scripts SQL para popular as tabelas fazem parte da questão).

- Na função executar_acao(), conectar-se ao banco de dados;
- Se a ação for mover, verificar se o jogador pode mover para a direção desejada, e atualizar a posicao no cenário atual;
- Se a ação for pegar bloco, verificar se há um bloco na direção desejada e, caso positivo, verificar se o jogador atende ao pré-requisito para pegar aquele tipo de bloco (Ex: possui a picareta). Se positivo, colocar o bloco em seu inventário e remover sua referência no cenário atual;
- Se a ação for largar bloco selecionado, verificar se a direção escolhida está livre e remover o bloco do inventário posicionando-o no cenário atual;
- Se a ação for transformar, a referência de objeto será de uma caixa mágica, tipo especial de objeto constituindo-se de uma ou um par de referências a itens de inventário do próprio jogador (Os itens são colocados e removidos da caixa mágica por outra ação auxiliar não abordada aqui, bastando preencher diretamente o script para teste nesta questão). A tranformação deve estar contida na tabela de receitas, e portanto deve ser uma receita válida. A receita não deve consumir mais ingredientes do que necessário para a mesma, atualizando as quantidades de blocos no inventário, além de criar um novo bloco/acessório no inventário em caso de sucesso;
- Após qualquer das ações acima executadas com sucesso, o programa deve registrar o comando na tabela de histórico, para fim de reconstruir a partida posteriormente. Caso a ação não possa ser executada, a função deve retornar false.

R: O cenário imaginado para esta questão, utilizado para popular as tabelas, é o retratado na seguinte figura:



O eixo coordenado começa na posição (0,0,0), na qual se encontra um bloco do tipo pedra. Mais alguns blocos do tipo pedra compõem uma superfície sobre a qual se encontra o personagem do jogador 1, mais precisamente na posição (1,2,1). Dois blocos de madeira estão empilhados um sobre o outro nas posições (0,0,1) e (0,0,2).

Scripts SQL para pré-população dos dados:

```sql
--
-- Dados da tabela `acao`
--

INSERT INTO `acao` (`Id`, `Descricao`) VALUES
(1, 'andar'),
(2, 'pegar bloco'),
(3, 'largar bloco'),
(4, 'transformar');

--
-- Dados da tabela `bloco`
--INSERT INTO `bloco` (`Id`, `Descricao`, `Tipo`,
`IdBloco_Prerequisito`) VALUES
(1, 'madeira', 'material', NULL),
(2, 'pedra', 'material', NULL),
(3, 'cabo madeira', 'material', NULL),
(4, 'marreta', 'acessorio', NULL),
(5, 'ferro', 'material', 3),
(6, 'picareta', 'acessorio', NULL),
(7, 'aco', 'material', 6),
(8, 'espada', 'acessorio', NULL);

--
-- Dados da tabela `cenario`
--INSERT INTO `cenario` (`Id`) VALUES
(1),
(2);

--
-- Dados da tabela `cenario_bloco`
--INSERT INTO `cenario_bloco` (`IdCenario`, `IdBloco`, `X`, `Y`, `Z`)
VALUES
(1, 1, 0, 0, 1),
(1, 1, 0, 0, 2),
(2, 1, 0, 0, 1),
(2, 1, 0, 0, 2),
(1, 2, 0, 0, 0),
(1, 2, 0, 1, 0),
(1, 2, 0, 2, 0),
(1, 2, 1, 1, 0),
(1, 2, 1, 2, 0),
```

```
(1, 2, 2, 2, 0),
(2, 2, 0, 0, 0),
(2, 2, 0, 1, 0),
(2, 2, 0, 2, 0),
(2, 2, 1, 1, 0),
(2, 2, 1, 2, 0),
(2, 2, 2, 2, 0);

--
-- Dados da tabela `jogador`
--INSERT INTO `jogador` (`Id`, `Conta`, `Email`, `Senha`,
`Aparencia`, `Nome_Personagem`, `Nivel`) VALUES
(1, 'fulano', 'fulano@mail', 'aSAKGLGJ', 1, 'FulanoBriks', 1);

--
-- Dados da tabela `jogador_cenario`
--INSERT INTO `jogador_cenario` (`IdJogador`, `IdCenario`, `X`, `Y`,
`Z`) VALUES
(1, 1, 1, 2, 1),
(1, 2, 1, 2, 1);

--
-- Dados da tabela `partida`
--INSERT INTO `partida` (`Id`, `IdCenarioInicial`, `IdCenarioAtual`)
VALUES
(1, 1, 2);

--
-- Dados da tabela `receita`
--INSERT INTO `receita` (`Id`, `IdBloco1`, `IdBloco2`,
`Quantidade_Bloco1`, `Quantidade_Bloco2`, `IdBlocoGerado`) VALUES
(1, 1, NULL, 10, NULL, 3),
(2, 2, 3, 10, 1, 4),
(3, 3, 5, 1, 10, 6),
(4, 7, NULL, 15, NULL, 8);

INSERT INTO `maquinamagica` (`IdJogador`, `IdBloco1`,
`QuantidadeBloco1`, `IdBloco2`, `QuantidadeBloco2`) VALUES
('1','1','10',NULL,NULL);
```

Código da questão:

```
<?php
```

```php
$id_cenario_atual = 2; //Exemplo

function conecta_banco()
{
   mysql_connect('localhost', 'root', '') or die('Impossivel conectar com banco.');
   mysql_select_db('ad2_s12013') or die('Schema nao existe no banco.');
}

function executar_acao($id_jogador, $id_acao, $id_bloco, $dx, $dy, $dz)
{
   global $id_cenario_atual;

   conecta_banco();

   switch ($id_acao) {
      case 1: //mover
         if ($dx == 0 && $dy == 0 && $dz == 0)
            return;
         //Poderia fazer uma unica query para performance, sem legibilidade.
         //1. Recuperar coordenadas do jogador no cenario atual

         $result  = mysql_query("SELECT X,Y,Z FROM Jogador_Cenario WHERE idJogador =
$id_jogador AND idCenario = $id_cenario_atual");
         $jogador = mysql_fetch_object($result);

         //2. Guardar novas posicoes
         $novo_X = $jogador->X + $dx;
         $novo_Y = $jogador->Y + $dy;
         $novo_Z = $jogador->Z + $dz;
         //3. Verificar se ha bloco ou outro jogador na posicao desejada
         $result = mysql_query("SELECT 1 FROM (SELECT X,Y,Z FROM cenario_bloco UNION
SELECT X,Y,Z FROM jogador_cenario) " . " e WHERE idCenario = $id_cenario_atual AND e.X =
$novo_X AND e.Y = $novo_Y AND e.Z = $novo_Z");

         if (!$result) {
            //4. Insere a acao no historico de acoes
            mysql_query("INSERT INTO acao_jogador_cenario (`IdJogador`, `IdCenario`, `IdAcao`,
`DX`, `DY`, `DZ`) VALUES ($id_jogador, $id_cenario_atual, 1,  $dx, $dy, $dz);");
            //5. Atualiza a posicao
            mysql_query("UPDATE jogador_cenario SET X = $novo_X, Y = $novo_Y, Z = $novo_Z
WHERE idJogador = $id_jogador AND idCenario = $id_cenario_atual;");
```

```
        }
        break;

    case 2:
        if ($dx == 0 && $dy == 0 && $dz == 0)
            return false;
        //1. Recuperar coordenadas do jogador no cenario atual
        $result  = mysql_query("SELECT X,Y,Z FROM Jogador_Cenario WHERE idJogador =
$id_jogador AND idCenario = $id_cenario_atual");
        $jogador = mysql_fetch_object($result);

        //2. Guardar posicao do bloco desejado
        $novo_X = $jogador->X + $dx;
        $novo_Y = $jogador->Y + $dy;
        $novo_Z = $jogador->Z + $dz;

        //3. Verificar se o bloco existe e qual seu tipo
        $result     = mysql_query("SELECT b.id, b.tipo FROM cenario_bloco cb INNER JOIN
bloco b ON cb.IdBloco = b.Id " . " WHERE idCenario = $id_cenario_atual AND e.X = $novo_X
AND e.Y = $novo_Y AND e.Z = $novo_Z");
        $tipo_bloco = mysql_fetch_object($result);

        if ($tipo_bloco) {
            $id_bloco   = $tipo_bloco->id;
            $tipo_bloco = $tipo_bloco->tipo;
            //4. Insere a acao no historico de acoes
            mysql_query("INSERT INTO acao_jogador_cenario (`IdJogador`, `IdCenario`, `IdAcao`,
`IdBloco`, `DX`, `DY`, `DZ`) VALUES ($id_jogador, $id_cenario_atual, 2, $id_bloco, $dx, $dy,
$dz);");
            //5. Remove o bloco do cenario atual
            mysql_query("DELETE FROM cenario_bloco WHERE IdCenario = $id_cenario_atual
AND X = $novo_X AND Y = $novo_Y AND Z = $novo_Z");
            //6. Atualiza inventario, se o bloco ja existir, aumenta o contador
            $result = mysql_query("SELECT 1 FROM inventario WHERE IdJogador = $id_jogador
AND IdBloco = $id_bloco;");

            if ($result)
                mysql_query("UPDATE inventario SET Quantidade = Quantidade + 1 WHERE
IdJogador = $id_jogador AND IdBloco = $id_bloco;");
            else
                mysql_query("INSERT INTO inventario(IdJogador,IdBloco) VALUES ($id_jogador,
$id_bloco);");
```

```
        } else
            return false;
        break;

    case 3:
        if ($dx == 0 && $dy == 0 && $dz == 0)
            return false;
        //1. Recuperar coordenadas do jogador no cenario atual
        $result  = mysql_query("SELECT X,Y,Z FROM Jogador_Cenario WHERE idJogador =
$id_jogador AND idCenario = $id_cenario_atual");
        $jogador = mysql_fetch_object($result);

        //2. Guardar posicao onde o bloco sera despejado
        $novo_X = $jogador->X + $dx;
        $novo_Y = $jogador->Y + $dy;
        $novo_Z = $jogador->Z + $dz;

        //3. Verificar se ha bloco ou outro jogador na posicao desejada
        $result = mysql_query("SELECT 1 FROM (SELECT X,Y,Z FROM cenario_bloco UNION
SELECT X,Y,Z FROM jogador_cenario) " . " e WHERE idCenario = $id_cenario_atual AND e.X =
$novo_X AND e.Y = $novo_Y AND e.Z = $novo_Z");

        if (!$result) {
            //4. Insere a acao no historico de acoes
            mysql_query("INSERT INTO acao_jogador_cenario (`IdJogador`, `IdCenario`, `IdAcao`,
`IdBloco`, `DX`, `DY`, `DZ`) VALUES ($id_jogador, $id_cenario_atual, 3, $id_bloco, $dx, $dy,
$dz);");

            //5. Atualiza inventario, se o bloco possuir apenas 1 unidade, apagar, senao
decrementar quantidade
            $result     = mysql_query("SELECT quantidade FROM inventario WHERE IdJogador =
$id_jogador AND IdBloco = $id_bloco;");
            $quantidade = mysql_fetch_field($result, 0);
            if ($quantidade > 1)
                mysql_query("UPDATE inventario SET Quantidade = Quantidade - 1 WHERE
IdJogador = $id_jogador AND IdBloco = $id_bloco;");
            else
                mysql_query("DELETE FROM inventario WHERE IdJogador = $id_jogador AND
IdBloco = $id_bloco;");

            //6. Posiciona bloco no cenario
```

```
            mysql_query("INSERT INTO cenario_bloco(`IdCenario`,`IdBloco`,`X`,`Y`,`Z`) VALUES
($id_cenario_atual, $id_bloco, $novo_X, $novo_Y, $novo_Z);");


        } else
            return false;

        break;
    case 4:
        //Verifica a receita para a qual os igredientes da caixa magica sao indicados e se estao
em quantidade suficiente
        mysql_query("SELECT r.`IdBlocoGerado`, r.`Quantidade_Bloco1`,
r.`Quantidade_Bloco2`  FROM receita r, maquinamagica m WHERE m.`IdJogador` = 1 AND
m.`IdBloco1` = r.`IdBloco1` AND (r.`IdBloco2` IS NULL OR m.`IdBloco2` = r.`IdBloco2`) AND
m.`QuantidadeBloco1` >= r.`Quantidade_Bloco1` AND (r.`IdBloco2` IS NULL OR
m.`QuantidadeBloco2` >= r.`Quantidade_Bloco2`);");

        $receita = mysql_fetch_object($result);

        if ($receita) {
            $id_bloco_gerado = $receita->IdBlocoGerado;
            $qtd_bloco1      = $receita->Quantidade_Bloco1;
            $qtd_bloco2      = $receita->Quantidade_Bloco2;

            //4. Subtrai ingredientes da caixa magica
            mysql_query("UPDATE `maquinamagica` SET `QuantidadeBloco1` =
QuantidadeBloco1 - $qtd_bloco1 WHERE `IdJogador` = 'IdJogador';");
            if ($qtd_bloco2 != null)
                mysql_query("UPDATE `maquinamagica` SET `QuantidadeBloco2` =
QuantidadeBloco2 - $qtd_bloco2 WHERE `IdJogador` = 'IdJogador';");

            //5. Gera novo item no inventario
            mysql_query("INSERT INTO inventario(IdJogador, IdBloco) VALUES ($id_jogador,
$id_bloco_gerado);");

            //6. Registra acao
            mysql_query("INSERT INTO acao_jogador_cenario (`IdJogador`, `IdCenario`,
`IdAcao`) VALUES ($id_jogador, $id_cenario_atual, 4);");
        } else
            return false;
        break;
    default:
```

```
        return false;
    }
    mysql_close();
}


executar_acao(1, 1, null, 0, -1, 0);
executar_acao(1, 1, null, -1, 0, 0);
executar_acao(1, 2, null, 0, -1, 1);
executar_acao(1, 3, 1, 1, 0, 0);
executar_acao(1, 4, null, null, null, null);
```

Questão 3

Pesquise e responda:

1. Por que o PHP não possui uma função para a qualquer momento, redirecionar o usuário para a próxima página? E como isto pode ser feito?

R: Após a requisição HTTP, o PHP é interpretado e o HTML resultante de sua execução é então devolvido ao navegador da máquina cliente. Não há como executar comandos após a interpretação. Existem duas maneiras de se contornar: uma é através dos recursos disponíveis no objeto window Javascript, que possui a função redirect. A outra forma, caso possamos realizar o redirecionamento após a submissão de um formulário, basta colocar antes de qualquer coisa impressa o comando: `header("Location: $url_destino");` gerando assim um cabeçalho HTML que aponta para outra página.

2. Para que serve o uso conjunto das palavras reservadas “or die()”, exemplifique um possível uso.

R: A sintaxe “or” se trata do operador “OU” com menor precedência que o “||”, no sentido em que a segunda operação será executada se a primeira falhar. O comando `die()` funciona como uma interrupção definitiva do script em execução, podendo receber uma mensagem para impressão. Um exemplo clássico acontece durante a conexão com o banco de dados:

```
mysql_connect('localhost', 'root', '') or die('Impossivel conectar com banco.');
```

3. O script PHP gerado ao final da execução da página fica disponível à máquina cliente? Por que?

R: Não, PHP é uma linguagem server-side, e toda lógica roda apenas no lado do servidor. A única linguagem percebida pela máquina cliente são HTML e linguagens de script ou estilização client-side.